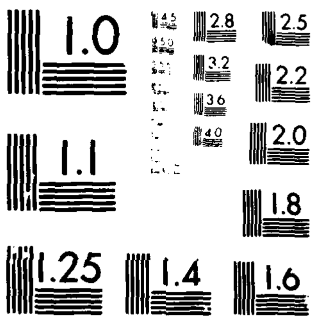
AD-A084 035   GEORGE WASHINGTON UNIV WASHINGTON D C PROGRAM IN LOG--ETC F/G 15/5
AN OPTIMIZATION TECHNICAL FOR A MULTITIME PERIOD SPARES PROVISI--ETC(U)
FEB 80 H K RAPPOPORT
UNCLASSIFIED SERIAL-T-408 NL

1 OF 2
AD
A084035



MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

LEVEL IV

ADA084035

THE
GEORGE
WASHINGTON
UNIVERSITY

INSTITUTE FOR MANAGEMENT SCIENCE AND ENGINEERING
SCHOOL OF ENGINEERING AND APPLIED SCIENCE

12

6

AN OPTIMIZATION TECHNIQUE FOR A MULTITIME PERIOD SPARES PROVISIONING PROBLEM

by

10 Harold [redacted] Rappoport pVI

Kalman

Serial T-408

11 7 Feb[redacted]80

12 116

14 SERIAL-T-4Ø8

The George Washington University
School of Engineering and Applied Science
Institute for Management Science and Engineering

15

Program ~~in Logistics~~
Contract N00014-75-C-0729
Project NR 347 020
Office of Naval Research

405 337 mt

NONE
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER T-408	2. GOVT ACCESSION NO. AD-A084034	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) AN OPTIMIZATION TECHNIQUE FOR A MULITITIME PERIOD SPARES PROVISIONING PROBLEM		5. TYPE OF REPORT & PERIOD COVERED SCIENTIFIC
		6. PERFORMING ORG. REPORT NUMBER T-408
7. AUTHOR(s) HAROLD K. RAPPOPORT		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-0729
9. PERFORMING ORGANIZATION NAME AND ADDRESS THE GEORGE WASHINGTON UNIVERSITY PROGRAM IN LOGISTICS WASHINGTON, DC 20052		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS OFFICE OF NAVAL RESEARCH CODE 434 ARLINGTON, VA 22217		12. REPORT DATE 7 FEBRUARY 1980
		13. NUMBER OF PAGES 110
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) NONE
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

THIS DOCUMENT HAS BEEN APPROVED FOR PUBLIC SALE AND RELEASE; ITS DISTRIBUTION IS UNLIMITED.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

SPARES PROVISIONING
NONLINEAR INTEGER PROGRAMMING
BRANCH AND BOUND
DYNAMIC PROGRAMMING
PARTIAL ENUMERATION
MACHINE REPAIR PROBLEM

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This work contains an algorithm that yields an exact solution to a spares provisioning problem that was "solved" previously using a heuristic algorithm. After presenting the backgrounds of the underlying gas turbine engine maintenance problem and the heuristic approach, a discussion of the techniques developed here to solve the problem exactly is presented. It is shown that the problem can be set in a form amenable to solution by a branch and bound procedure. In order to solve the subproblems, it is

(Continued)

20. Abstract - continued

necessary to characterize the constraint region for each time period. An enumeration scheme, together with the concept of a "corner" is introduced to accomplish this. Finally, dynamic programming is used to obtain the optimum of each subproblem associated with the branch and bound approach.

Computational results are given for four problems. Three of these problems are modifications of the basic problem; they are presented to demonstrate the nature of the solution technique. The fourth problem is identical to a problem previously solved with the heuristic algorithm. Extensions to the original problem are then discussed.

THE GEORGE WASHINGTON UNIVERSITY
School of Engineering and Applied Science
Institute for Management Science and Engineering

Program in Logistics

Abstract
of
Serial T-408
7 February 1980

AN OPTIMIZATION TECHNIQUE FOR A MULTITIME PERIOD SPARES PROVISIONING PROBLEM

by

Harold K. Rappoport

This work contains an algorithm that yields an exact solution to a spares provisioning problem that was "solved" previously using a heuristic algorithm. After presenting the backgrounds of the underlying gas turbine engine maintenance problem and the heuristic approach, a discussion of the techniques developed here to solve the problem exactly is presented. It is shown that the problem can be set in a form amenable to solution by a branch and bound procedure. In order to solve the subproblems, it is necessary to characterize the constraint region for each time period. An enumeration scheme, together with the concept of a "corner," is introduced to accomplish this. Finally, dynamic programming is used to obtain the optimum of each subproblem associated with the branch and bound approach.

Computational results are given for four problems. Three of these problems are modifications of the basic problem; they are presented to demonstrate the nature of the solution technique. The fourth problem is identical to a problem previously solved with the heuristic algorithm. Extensions to the original problem are then discussed.

Research Supported by

Contract N00014-75-C-0729
Project NR 347 020
Office of Naval Research

AN OPTIMIZATION TECHNIQUE FOR A MULTITIME PERIOD SPARES PROVISIONING PROBLEM

By

Harold Kalman Rappoport

B.S. 1965, Stevens Institute of Technology

M.S. 1970, University of Maryland

A Dissertation submitted to

The Faculty of

The School of Engineering and Applied Science

of The George Washington University in partial satisfaction

of the requirements for the degree of Doctor of Science

February 18, 1980

Dissertation directed by

James Edward Falk

Professor of Operations Research

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Abstract

AN OPTIMIZATION TECHNIQUE FOR A MULTITIME

PERIOD SPARES PROVISIONING PROBLEM

By

Harold Kalman Rappoport

James Edward Falk, Director of Research

This work contains an algorithm that yields an exact solution to a spares provisioning problem that was "solved" previously using a heuristic algorithm. After presenting the backgrounds of the underlying gas turbine engine maintenance problem and the heuristic approach, a discussion of the techniques developed here to solve the problem exactly is presented. It is shown that the problem can be set in a form amenable to solution by a branch and bound procedure. In order to solve the subproblems, it is necessary to characterize the constraint region for each time period. An enumeration scheme, together with the concept of a "corner" is introduced to accomplish this. Finally, dynamic programming is used to obtain the optimum of each subproblem associated with the branch and bound approach.

Computational results are given for four problems. Three of these problems are modifications of the basic problem; they are presented to demonstrate the nature of the solution technique. The fourth problem is identical to a problem previously solved with the heuristic algorithm. Extensions to the original problem are then discussed.

ACKNOWLEDGMENT

If I were to acknowledge individually all of those who have helped and encouraged me throughout this long effort, this acknowledgment might be longer than the text of this dissertation. These people include my employers, who supported me; the faculty of The George Washington University, who educated me; my family, who encouraged me; and my friends, who tolerated me.

Of course there are those who deserve special thanks. First I would like to thank Mrs. Gail Clark, who spent many hours of her own time typing the innumerable versions of the first draft of this dissertation, and Summit Research Corporation, the use of whose facilities and the encouragement of whose officers and employees lightened this burden.

I am particularly indebted to Professor James Falk for his role as both a teacher and an advisor over the many years. His patience and tolerance have been greatly appreciated. Also, I thank Professor Donald Gross, who suggested the dissertation topic and offered guidance and assistance throughout this effort.

Most of all I thank my family: my parents, my wife, and my children. I thank my parents for instilling in me their values that made me persevere through this arduous task. I thank my wife, Ellen, for encouraging and assisting me, for being a single parent on many a winter's night, for taking out the garbage on Tuesday nights, and for accepting all those other responsibilities that I did not have time to assume myself. Last but not least, I thank my daughters, Amy and Erica, for their cooperation and for the sacrifices they unknowingly made as I pursued this goal.

CHAPTER I

INTRODUCTION

This dissertation develops a technique for optimizing a spares provisioning problem over a multiyear period. The spares provisioning problem has been previously formulated [1] and a heuristic optimization procedure specified. The complexity of the problem is such that it could not be solved exactly by standard techniques at that time. Through the use of an amalgamation of branch and bound, partial enumeration, and dynamic programming techniques, an approach that assures convergence to the optimal solution is obtained here. This work discusses and justifies the solution technique to the spares provisioning problem.

I.1 Background

The spares provisioning problem is a stochastic problem in which an optimum policy for purchasing spare equipment and repair channels over a multitime period horizon is sought. For each time period, a specified number of machines are required to be on line. As normally happens, some of these machines will break down. Upon breakdown, a machine is removed to a repair station and replaced by a spare, if one is available. To maintain a reasonable level of operation there is a requirement placed upon the system that a spare be available a specified percentage of the time.

There is a tradeoff to be made between the number of spares purchased and the number of repair facilities set up. If a machine must wait to be repaired because all of the repair facilities are busy, that machine is not available for use as a spare for a longer period of time. The greater the number of repair facilities, the greater the likelihood that a machine will not wait to be repaired. On the other hand, if there is an abundance of spare machines available, a delay in the repair of the machine may not impact on the system. It is desired, therefore, depending on the cost of the spares, the cost of setting up a repair facility, and the relative rates of machine failure and repair, to determine the most cost effective mix of spares and repair facilities that satisfies the availability requirement.

The problem is further complicated because the optimization is over multiple time periods. At the beginning of each time period the number of machines on line may be changed. Either new machines may be added or old machines removed. Because this problem is over a multiperiod horizon, the optimal policy of selecting spares or repair channels for each individual time period is not necessarily the optimal policy when considered in the context of the total horizon. Repair channels and spares that may not be needed in the current time period but that will be required in a future time period may be purchased in the current period because of lower cost.

From the above description of the problem it might appear that obtaining a solution should not be too difficult. It is a multiperiod problem for which there are two independent variables (number of spares and number of repair channels) for each time period. There is also a single constraint imposed on the availability of replacement machines for each of the time periods and an objective function associated with the purchase cost of spares and machines. If the objective and constraining functions had reasonably simple structures, the problem could be solved using standard integer linear or nonlinear solution techniques. The difficulty, however, is that the constraint equations do not have a nice functional dependence on the independent variables. Because of the stochastic nature of the problem, there is a complicated dependence on the variables.

The underlying stochastic process can be formulated in terms of the classical machine repair problem. The work of Taylor and

Jackson [2] was one of the earliest applications of queueing theory to the spares provisioning problem. In this formulation, the service time and the time between failures are assumed to be exponentially distributed random variables. As will be shown subsequently, the steady state probability that n machines are in need of repair can be specified as a function of the number of machines in the system, the number of repair channels, the number of spares, and the relative failure rate and repair rate. The explicit formulation is dependent on whether the number of repair channels is greater or less than the number of spares and whether n , the number in repair, is greater or less than the number of channels or number of spares. As will be seen, it is this aspect of the problem that causes the greatest complication since the explicit forms of the constraints cannot be specified until the policy is known.

Another facet of the problem, which is a result of the multiple time periods, is the determination of the machine failure rate. As part of the formulation of the problem (see Reference [1]), the failure rate specified for each time period is the average failure rate taken over the number of machines introduced in the time period, the number of machines repaired in the previous time period, and the machines from the previous time period that had not been repaired. In that the expected number of machines repaired in a time period is dependent on the number of repair channels, the average failure rate for a given time period is dependent on the number of repair channels in previous time periods.

Therefore, what may at first appear to be a simple problem has become very complex in that the functional dependence is not explicitly known until a policy is enumerated. and a key parameter, the failure rate, is dependent on the selected policy of previous time periods. Because of these two complicating factors a solution technique that combined partial enumeration and the sequential technique of branch and bound was needed to solve the problem.

I.2 Purpose

Spares provisioning analysis has been applied to the maintenance of marine gas turbine engines for the U.S. Department of the Navy [1]. For this particular problem the reliability of a new gas turbine engine is assumed to depend on its time of introduction. The reliability of a used item is assumed to depend on the time of its last repair. It is assumed that there is a learning curve, and that with increased experience, the gas turbine engines will have an improved reliability. Therefore, the failure rate used in the formulation of this problem is dependent on the number of engines introduced in each time period as well as on the number repaired in previous time periods. Through this failure rate the availability constraint for a given time period is coupled to the policy used in all of the previous time periods. This kind of situation may be typical of many repairable item inventory systems in which the items to be controlled result from new technology.

The determination of an optimal purchase policy for the mix of spares and repair channels needed in each time period was previously attempted using a heuristic approach which separated the problem by time periods [1]. While this approach provided an apparently "good" policy--and a feasible one in terms of its satisfaction of the spares availability constraint--there was no assurance that it was optimal, and it has been shown to be sub-optimal for some cases.

This dissertation employs the same basic model as was used in [1], but approaches the optimization in a different manner. The reliability for each time period is initially approximated by ignoring the terms dependent on the previously repaired components, and an optimum policy for the approximate problem is determined over all time periods. The resulting trial solution is checked for feasibility. If the solution is not feasible, branching occurs, creating a new set of reliability factors. The new set of problems, using the updated reliability factors, more closely bounds the original problem. This branch and bound technique proceeds until the optimal solution is found.

The use of this approach still results in an integer programming model. Through techniques discussed within this paper, the lattice of integers that define the boundaries of the constraint set are determined. By taking the convex hull of this constraint set and using a linear objective function, the problem could then be solved using the simplex method. However, it was found to be computationally simpler to solve the problem using dynamic programming.

The technique described in this dissertation not only gives an optimal solution to the spares provisioning problem, but also requires less computer capability and is computationally faster than the previous heuristic approach. Furthermore, the method is applicable to a more general type of problem and can be used to solve the particular spares provisioning problem with a more complex cost function.

I.3 Overview of the Study

This work consists of four additional chapters plus two appendices. Chapter II presents the spares provisioning problem in detail, with an explicit mathematical characterization. The heuristic solution suggested in [1] is also discussed. In Chapter III the mathematical development of the optimization techniques is discussed. This includes a discussion of the convergence of the solution, the technique for the generation of the constraint set, and the dynamic programming solution. Computational results and example problems with solutions are presented in Chapter IV. The final chapter summarizes this study and presents conclusions and recommendations.

CHAPTER II

PROBLEM DEFINITION

While most of the effort described in this dissertation has dealt with the solution of a particular problem, an optimal spares provisioning policy for maintaining an adequate number of operating marine gas turbine engines, the techniques developed herein can be applied to a more general class of problems. This chapter discusses the particular gas turbine engine problem, its formulation, and a previously specified heuristic approach for obtaining a good policy.

II.1 The Gas Turbine Engine Problem

The immediate problem addressed here is one of determining the minimum cost policy for purchasing spares and repair channels that assures that there is an adequate supply of spares available for replacing marine gas turbine engines* that randomly fail. A multiyear planning horizon is considered, with the possibility that additional turbines may be introduced in each year, requiring additional spares and/or repair channels.

When a turbine fails, a request is made for a replacement. This replacement comes from the pool of spares, and the component that failed is removed and taken to the repair facilities. If at least one repair channel is not busy, repair begins immediately;

*There are two components in a gas turbine: the gas generator and the power turbine. Each component type requires its own individual set of spares and repair channels, and thus we can consider independent provisioning systems. We will use the term "turbine," but in reality we are dealing with either the gas generator or the power turbine components.

if all facilities are busy, the turbine joins the queue and waits for service. Upon completion of the repairs, the gas turbine is placed into the spares pool. If there is an inadequate number of repair facilities, a queue of nonfunctioning turbines will develop; as the queue builds and the spares pool diminishes, it is possible for the spares pool to become depleted. If a failure occurs when the spares pool is empty, there will be no available replacement and the ship in which the failed turbine resides will not be able to function. Therefore, there is imposed on the problem the constraint that the probability that a spare is available when a turbine fails be at least 0.90 (or any other value specified).

Through either a learning process or a conscious component improvement program (CIP), it is assumed that the reliability of a turbine improves both with successive years of manufacture, and after each repair. In conjunction with these assumptions is the assumption that the mean failure rate for a time period is the average of the failure rate for the new machines introduced in the time period, the failure rate for the turbines repaired in the previous time period, and the mean failure rate assumed for the previous time period for those turbines that were not repaired in that time period.

This problem was formulated in [1] as the "classical" machine repair with spares queueing problem. Failure and service times are assumed to be exponentially distributed random variables. The formulation of Reference [1] is presented here. Based upon this framework, a mathematical programming problem is defined. The solution technique for this problem is given in the next chapter.

II.2 Queueing Model

In the classical machine repair with spares problem, the machines may be in one of four distinct states. They may be up and operating; down and in the queue awaiting repair; undergoing repair; or in the spares pool, where they are available as replacements.

For a given set of parameters (the service rate, repair rate, number of machines, number of spares, and number of repair channels) under the assumption of exponential service and failure distributions, the system will eventually reach a steady state; that is, the probability that a machine is in a particular state is independent of time. Since the steady state solution of the classical machine repair with spares problem is known and the transient model appears to be analytically intractable, steady state is assumed in this dissertation. In reality, the problem never reaches steady state since new gas turbines are introduced at random intervals of time throughout the year. However, it is assumed that new turbines are introduced only at the beginning of each year and that the transient effects caused by the introduction of turbines damp out immediately. Therefore, each time period analyzed uses the steady state formulation.

The transient effects are currently under investigation, but since this dissertation is concerned with the optimization of the purchase policy, the steady state assumption is accepted without further discussion [3].

To understand the formulation of this problem, the following notation is introduced.

$p_{n,i}$	= steady-state probability that n items are "down" in year i
c_i	= number of repair facilities to plan for year i
y_i	= number of spares to plan for year i
λ_i	= component failure rate (Poisson mean) for new or repaired items in year i
$\overline{\lambda}_i$	= average failure rate for the population in year i
$\overline{R}_i$	= expected number of components repaired for year i
M_i	= component population size in year i
$1/\mu_i$	= average service (removal, transportation, and repair) time, exponential mean, for year i .

For steady state, the following equations are obtained [1]:

$$p_{n,i} = \begin{cases} \dfrac{M_i^n}{n!}\left(\dfrac{\overline{\lambda}_i}{\mu_i}\right)^n p_{0,i} & (0 \leq n \leq c_i) & & a \\[2ex] \dfrac{M_i^n}{c_i^{\,n-c_i}\; c_i!}\left(\dfrac{\overline{\lambda}_i}{\mu_i}\right)^n p_{0,i} & (c_i \leq n \leq y_i) & \langle c_i \leq y_i \rangle & b \\[2ex] \dfrac{M_i^{y_i} M_i!}{(M_i-n+y_i)!\; c_i^{\,n-c_i}\; c_i!}\left(\dfrac{\overline{\lambda}_i}{\mu_i}\right)^n p_{0,i} & (y_i \leq n \leq y_i+M_i) & & c \\[2ex] \dfrac{M_i^n}{n!}\left(\dfrac{\overline{\lambda}_i}{\mu_i}\right)^n p_{0,i} & (0 \leq n \leq y_i) & & d \\[2ex] \dfrac{M_i^{y_i} M_i!}{(M_i-n+y_i)!\; n!}\left(\dfrac{\overline{\lambda}_i}{\mu_i}\right)^n p_{0,i} & (y_i \leq n \leq c_i) & \langle c_i \geq y_i \rangle & e \\[2ex] \dfrac{M_i^{y_i} M_i!}{(M_i-n+y_i)!\; c_i^{\,n-c_i}\; c_i!}\left(\dfrac{\overline{\lambda}_i}{\mu_i}\right)^n p_{0,i} & (c_i \leq n \leq y_i+M_i) & & f \end{cases} \tag{II-1}$$

$$p_{0,i} : \sum_{n=0}^{M_i+y_i} p_{n,i} = 1$$

$$\overline{\lambda}_i = \begin{cases} \left\{(M_i - M_{i-1})\lambda_i + \overline{R}_{i-1}\lambda_{i-1} + (M_{i-1} - \overline{R}_{i-1})\overline{\lambda}_{i-1}\right\} / M_i , & M_i \geq M_{i-1} \\ \left\{\overline{R}_{i-1}\lambda_{i-1} + (M_{i-1} - \overline{R}_{i-1})\overline{\lambda}_{i-1}\right\} / M_{i-1} , & M_i \leq M_{i-1} \end{cases} \tag{II-2}$$

$$\overline{R}_i = \overline{\lambda}_i \left[M_i - \sum_{n=y_i+1}^{M_i+y_i} (n-y_i) p_{n,i} \right] . \tag{II-3}$$

A review of these equations gives insight into the nature of the problem. Equation (II-1) gives the probability that the system is in state n for the year i ; a state is defined by the number of machines that are down. The form of the expression for $p_{n,i}$ is dependent on the relative values of n_i , M_i , c_i , and y_i . Equation (II-1) consists of six different expressions. The first three (*a*,*b*,*c*) hold if there are at least as many (if not more) spares as there are repair channels. The second three expressions (*d*,*e*,*f*) hold if the number of repair channels is greater than the number of spares. The exact expression for $p_{n,i}$ is dependent on whether n is greater than, less than, or equal to c_i or y_i . It is apparent, therefore, that it is not possible to compute the state of the system, as determined by the probability that n machines are down, until a history of the number of spares and repair channels is specified. This aspect of the "undetermined form of $p_{n,i}$ " is a major factor that complicates the obtaining of a solution to the problem.

The term $p_{0,i}$ is the probability that no machines are down in time period i . Since this term is a common factor in all expressions of $p_{n,i}$, it acts as a normalizing factor and assures that the sum of all probabilities over all states equals one.

Equation (II-2) gives the mean failure rate $\overline{\lambda}_i$ for two cases, one when the number of machines in the system is increasing, and one when they are decreasing. When the population is increasing $(M_i \geq M_{i-1})$, the expression is composed of three terms. The first reflects the failure rate of all components introduced in time period i , and the second term reflects the change in failure rate due to those components that failed and were repaired in the previous time period $(i-1)$. The last term is associated with those components that were neither introduced in time period i nor repaired in time period $i-1$; such components maintain the previous period's failure rate. It is assumed that components that are either introduced or repaired in a time period i attain the same failure rate. The expression for the mean failure rate when the population is decreasing $(M_i \leq M_{i+1})$ is similar to the expression for an increasing population except for the first term. Since there are no machines being introduced in time period i , there is no dependence on the reliability of new machines.

The use of a weighted mean for the expected failure rate was investigated in References [4] and [5]. It was found that this yields conservative results when the availability is calculated. The validity of the approximation decreases as the failure rate increases. In the use of the approximation, it is assumed that no machine fails more often than once in a time period.

Equation (II-3) gives the mean number of machines repaired in a time period.

Next, the constraint on the availability of spares must be specified. This requires the calculation of the failure point probabilities, q_n , which are the state probabilities determined at the instant of failure. The state probability, p_n , is the probability that n machines are down at any arbitrary time. The q_n's , on the other hand, are measured only at the instant of failure. Between failures it is possible for a machine to be repaired, thereby changing the state of the system. The failure point probabilities, therefore, are the state probabilities, conditioned on the instant of failure; or, using the standard terminology of the literature,

$$q_n \equiv \Pr[n \text{ in system} \mid \text{a failure is about to occur}] .$$

Since the constraint on the availability of spares requires that a spare gas turbine be available when requested with a probability of 0.90, and that the request be made at the time of failure, the availability constraints are formulated in terms of the q_n's .

From Bayes' theorem,

$$q_n = \frac{\Pr[n \text{ in system}] \Pr[\text{failure about to occur} | n \text{ in system}]}{\sum_n [\Pr\{n \text{ in system}\} \Pr\{\text{failure about to occur} | n \text{ in system}\}]} .$$

For this problem, $q_{n,i}$ is determined to be

$$q_{n,i} = \begin{cases} \dfrac{M_i p_{n,i}}{M_i - \sum\limits_{\ell=y_i}^{y_i+M_i} (\ell - y_i) p_{\ell,i}} , & (0 \leqq n < y_i) \\ \dfrac{(M_i - n + y_i) p_{n,i}}{M_i - \sum\limits_{\ell=y_i}^{y_i+M_i} (\ell - y_i) p_{\ell,i}} , & (y \leqq n \leqq y_i + M_i) \end{cases} , \quad \text{(II-4)}$$

where the $p_{n,i}$ are given in Equation (II-1). Therefore, the availability constraint is specified as

$$\sum_{n=0}^{y-1} q_{n,i} \geqq 0.90 , \quad i=1,2,\ldots,k ,$$

where

$q_{n,i}$ = the steady state probability that n machines are down at the time of a machine failure, year i ; and

k = length of the planning horizon.

II.3 Cost Model

The problem as originally stated in Reference [1] had a cost model that was dependent on four types of costs: purchase cost of spares, purchase cost of repair channels, repair costs, and investment costs for component improvement. This cost function is expressed as:

$$Z = \sum_{i=1}^{k} d^i \left[C_{1,i}(c_i - c_{i-1})^+ + C_{2,i}(y_i - y_{i-1})^+ + C_{3,i}\bar{R}_i + C_{4,i} \right] ,$$

where

$(a)^+ = \max(a,0)$

$C_{1,i}$ = purchase cost of a repair channel, year i

$C_{2,i}$ = purchase cost of a spare component, year i

$C_{3,i}$ = cost of repairing a component, year i

$C_{4,i}$ = investment in reliability growth (CIP) program, year i

d = discount factor

k = number of years in planning horizon.

The mathematical programming problem becomes

$$\min_{c_1,c_2,\ldots,c_k;\ y_1,y_2,\ldots,y_k} Z = \sum_{i=1}^{k} d^i \left[C_{1,i}(c_i - c_{i-1})^+ + C_{2,i}(y_i - y_{i-1})^+ + C_{3,i}\bar{R}_i + C_{4,i} \right] \tag{II-5}$$

$$\text{Subject to} \quad \sum_{n=0}^{y_i - 1} q_{n,i} \geq 0.90 , \quad (i=1,2,\ldots,k) \tag{II-6}$$

$$c_i \geq 0 \text{ ; integer}$$

$$y_i \geq 0 \text{ ; integer.}$$

Since the last term, $C_{4,i}$, is constant for each time period, it can be dropped from the optimization. The third term, which is dependent on $\bar{R}_i$, is an explicit function of y_i and an implicit function of c_i and y_i . In the original analysis of

Reference [1] this term is dropped from the cost function. Since $\bar{R}$ is relatively constant [if (II-6) is satisfied], this term will only have a secondary effect on optimality. The optimization problem then becomes

$$\min_{c_i, y_i} \quad Z = \sum_{i=1}^{k} d^i \Big[C_{1,i}(c_i - c_{i-1})^+ + C_{2,i}(y_i - y_{i-1})^+ \Big]$$

$$\text{subject to} \quad \sum_{n=0}^{y_i - 1} q_{n,i} \geqq 0.90 \,, \quad \text{for all } i$$

$$c_i \geqq 0 \; ; \text{ integer}$$

$$y_i \geqq 0 \; ; \text{ integer.}$$

Since there is no cost benefit associated with selling a spare or repair channel, the problem can be reformulated to have a linear objective function, which may be written as follows:

$$\min_{c_1, c_2, \ldots, c_k ;\, y_1, y_2, \ldots, y_k} \quad Z = \sum_{i=1}^{k} d^i \Big[C_{1,i}(c_i - c_{i-1}) + C_{2,i}(y_i - y_{i-1}) \Big]$$

$$\text{subject to} \quad \sum_{n=0}^{y_i - 1} q_{n,i} \geqq 0.90 \,, \quad \text{for all } i$$

$$\begin{aligned} & c_i \quad \text{integer} \\ & c_i \geqq c_{i-1} \\ & c_1 \geqq 0 \\ & y_i \quad \text{integer} \\ & y_i \geqq y_{i-1} \\ & y_1 \geqq 0 \,. \end{aligned} \qquad \text{(II-7)}$$

This is the basic programming problem whose solution is discussed in this paper. It is a nonlinear integer programming problem with a linear objective function. A solution is achieved here using an amalgamation of partial enumeration and branch and bound, and dynamic programming techniques. The techniques developed can be applied to a more general objective function as well as to a more general problem type, which will be discussed in the final chapter.

II.4 Heuristic Approach

An attempt to solve this problem using a heuristic approach was presented in [1]. This approach is predicated on a procedure that minimizes the incremental cost of purchasing spares or repair channels for each successive time period. Starting with the number of spares and repair channels already purchased in the previous time period, the algorithm searches for the policy that satisfies the availability constraint for the current time period for a minimal additional cost. If in a time period the availability constraint is satisfied by the policy of the previous time period, a reduction procedure is employed that tries to decrease the number of spares or repair channels in such a manner that the availability constraint is merely satisfied, even though there is no explicit cost savings. External manipulation of the input to the algorithm allows modification of the policy used in previous time periods based upon the policy obtained for later time periods.

Based upon a comparison of computational experience, this algorithm provides results that approach the optimal policy, but does not assure that the optimal policy will be obtained. Its basic weakness is its one year at a time "optimality," which does not foresee future cost savings. The computational results are presented in Chapter IV of this dissertation.

CHAPTER III

METHODOLOGY

III.1 Introduction

In this chapter the methodology developed to solve this spares provisioning problem is presented. There are many unique aspects of the problem that require individual solution techniques; the difficulties created by these aspects and the techniques required to overcome them are discussed here. The overall solution technique and proof of the validity of the method are presented as well.

Contained in this chapter are: a discussion of each of the problematic areas, an overview of the solution technique, the method for generating regions that satisfy the availability constraint for each time period, the method of combining these regions across time periods to form the feasible space, the dynamic programming solution, and a discussion of the branch and bound algorithm utilized.

III.2 Problematic Areas

The most obvious difficulty of this problem is that it is a nonlinear integer program. A second difficulty is associated with the complicated functional dependence in the equations defining the problem. The last major complication is that a key parameter cannot be predicted until the optimal policy is specified.

The nonlinear integer aspect of the problem is obvious. While the objective function can be restructured into a linear form by adding the proper constraints, the nonlinearity of the state $p_{n,i}$ affects the determination of the constraint equations. As will be discussed later, the linearity of the objective function is important in the solution technique.

The form of the equation of state as seen in Equation (II-1) requires an enumerative approach to the solution. Unless the values of c, the number of repair channels, and y, the number of spares, are known relative to M (the number of turbines on line) and n (the number of turbines down), it is not possible to specify the form or value of p_n, the probability that n turbines are down. However, for any given "policy" $(c_1,c_2,\ldots, y_1,y_2,\ldots)$, it is possible to evaluate $p_{n,i}$ for all n.

The remaining difficulty arises from the nature of $\overline{\lambda}_i$, the average failure rate in time period i. As used in Equation (II-1), it appears that $\overline{\lambda}_i$ is a parameter that is required to determine the state variables $p_{n,i}$. Actually, $\overline{\lambda}_i$ is an implicit function of c_j and y_j for j less than i. With $\overline{\lambda}_i$ dependent, bounding the feasible region in time period i becomes extremely difficult, if not impossible. In using a branch and bound procedure, $\overline{\lambda}_i$ is approximated in a manner that makes it independent of the (c_i,y_i) choices of the previous time periods. By employing such a procedure it is possible to specify the convex hull of the region that satisfies the availability constraints for each time period independent of the other time periods.

Critical to the approach used for solving this problem is an understanding of the nature of the solution to problems with linear objective functions. Theorem 2 of Reference [6] states that:

". . . a linear objective function assumes its minimum at an extreme point of the convex set

> K generated by the set of possible solutions to the linear programming problem. If it assumes its minimum at more than one extreme point, then it takes on the same value for every convex combination of these particular points."

An extreme point of a convex set is a point in that set that cannot be expressed as a proper convex combination of any two distinct points in the set. The fact that this problem is not a linear programming problem in the normal sense does not invalidate the theorem above. The convex hull of the lattice of integer pairs satisfying the availability constraint could be determined and specified by a finite number of linear inequalities. Since the remaining constraints $(c_i \geq c_{i-1}, y_i \geq y_{i-1}, c_i \geq 0, y_i \geq 0)$ are linear, the problem could then be formulated as a linear integer programming problem.

By the above theorem, however, it is not necessary to go through the cumbersome technique of solving the linear programming problem. In the approach described herein, the extreme points of the convex hull of the lattice of integer pairs satisfying the availability constraint are determined explicitly through the application of a specified algorithm. Once these extreme points have been enumerated, the complete set of extreme points associated with the remaining linear constraints linking the time periods can also be listed. A dynamic programming approach is then used to determine the extreme point that gives the minimum of the objective function.

Therefore, the overall approach can be considered in terms of the three major techniques of which it is composed:

(1) a branch and bound procedure based upon the approximation of $\overline{\lambda}_i$. By approximating $\overline{\lambda}_i$, the dependence of $\overline{\lambda}_i$ on the c_j's and y_j's , for j less than i , can be eliminated from the

subproblems of the branch and bound technique. In this manner, $\overline{\lambda}_i$ becomes a parameter of the problem.

(2) A finite enumeration technique, made possible by a linear objective function, which allows the solution to lie on the extreme points of the convex hull. These extreme points are determined explicitly.

(3) Dynamic programming techniques, applied to obtain an optimum solution for the given set of $\overline{\lambda}_i$'s , once the set of extreme points of the convex hull has been determined.

The remainder of this chapter will be devoted to the enumeration technique and the proof that the procedure is finite, review the dynamic programming approach, and show that branch and bound is applicable to this problem. The branch and bound procedure is discussed next.

III.3 Branch and Bound

In determining if the availability constraint is satisfied at a particular (c_j, y_j) pair, it is necessary to evaluate $\overline{\lambda}_i$. Every $p_{n,i}$, the state probability for time period i , is dependent on $\overline{\lambda}_i$. It would be convenient if the $\overline{\lambda}_i$'s were parameters of the problem, but we know that they are dependent on the choice of policy in the previous time periods. The mean failure rate for a time period i is given by Equation (II-2) and repeated here:

$$\overline{\lambda}_i = \begin{cases} \left\{(M_i - M_{i-1})\lambda_i + \overline{R}_{i-1}\lambda_{i-1} + (M_{i-1} - \overline{R}_{i-1})\overline{\lambda}_{i-1}\right\} / M_i , & M_i \geq M_{i-1} \\ \left\{\overline{R}_{i-1}\lambda_{i-1} + (M_{i-1} - \overline{R}_{i-1})\overline{\lambda}_{i-1}\right\} / M_{i-1} , & M_i \leq M_{i-1} \end{cases} .$$

As was mentioned in Chapter II, this is an approximation to the failure rate associated with the process. Since the approximation deteriorates as $\overline{\lambda}_i$ increases, it has therefore been assumed that λ_i is small and that thus $\overline{\lambda}_i$ is also. An upper limit imposed on $\overline{\lambda}_i$ is given later in this section.

The dependence of $\overline{\lambda}_i$ on the policies of previous time periods is introduced through the $\overline{R}_i$, the expected number of machines repaired in the previous time period. Through its dependence on $p_{n,i}$, $\overline{R}_i$ is both explicitly dependent on y_i and implicitly dependent on y_i and c_i . If $\overline{R}_i$ were a parameter of the problem, $\overline{\lambda}_i$, for all i , could be specified. The branch and bound procedure is based upon the parameterization of $\overline{R}_i$.

The branch and bound procedure is predicated on the assumption that the initial problem may be decomposed into a set of subproblems with a subset of these problems solved in a series of stages [7].

Suppose the original problem is of the form

$$\min \; f(x)$$

$$\text{s.t.} \; x \in S ,$$

and X^* is the solution set. A branch and bound procedure initially addresses an extended problem of the form

$$\min \; F(x)$$

$$\text{s.t.} \; x \in T ,$$

where $S \subseteq T$ and $f(x) = F(x)$ for $x \in S$. A sequence of stages is created, the first stage defined by the initial extended problem above. Associated with a subsequent stage n is a collection of problems of the form

$$\min \; F(x)$$
$$\text{s.t.} \quad x \in T^i \,, \quad i \in N(n) \,.$$

Here N(n) defines the set of indices for the collection of problems composing stage n .

The branch and bound method will ensure that

$$\bigcup_{i\in N(n)} T^i \subset T \quad \text{and} \quad X^* \cap \left(\bigcup_{i\in N(n)} T^i \right) \neq \Phi$$

at each stage n ; i.e., each stage is comprised of *subproblems* whose collective feasible region contains at least one solution of the original problem. The solution to each of the i problems provides a lower bound $(LB)_i$ to f over $S \cap T^i$. The best lower bound at stage n , $(BLB)_n$, is the minimum of the set of lower bounds at stage n , written as

$$(BLB)_n = \min_{i\in N(n)} (LB)_i \,.$$

Since $S \subseteq T$ and $f(x) = F(x)$ for $x \in S$, the $(BLB)_n \leqq f(x^*)$, with $x^* \in X^*$. At each stage of the branch and bound procedure a new set of problems is solved. This set of problems is chosen in a manner such that

$$\bigcup_{i\in N(n+1)} T^i \subset \bigcup_{i\in N(n)} T^i \,.$$

Proceeding in this manner from stage to stage, successive best lower bounds are generated, leading to the following property of the branch and bound procedure:

$$(BLB)_1 \leqq \cdots \leqq (BLB)_n \leqq f(x^*) \,.$$

If at any stage of the problem a solution that gives the best lower bound at that stage is contained in S , the feasible region of the original problem, then this solution is also a solution to the original problem. Also, a lowest valued solution that is feasible to the original problem gives the best upper bound (BUB). If the BUB = BLB , the problem is solved with the solution value equal to the BUB = BLB .

In applying the branch and bound procedure to the spares provisioning problem, we use the objective function as given in (II-7). The feasible region of the extended problem is created by modifying the definition of $\overline{\lambda}_i$. By defining $\hat{\lambda}_i$ such that $\hat{\lambda}_i \leqq \overline{\lambda}_i$ for all feasible $\overline{\lambda}_i$ and for all i , we generate a region T such that $S \subseteq T$.

Theorem 1: For $\hat{\lambda}_i \leqq \overline{\lambda}_i$, a feasible region T associated with the problem given in (II-6) can be generated such that $S \subseteq T$, where S is the feasible region of problem (II-6).

Proof: This can be proven by considering the availability constraint. This is the only constraint that is affected by modifying $\overline{\lambda}_i$. Rewriting the availability constraint as in Appendix B, Equation (B-1), we have

$$\sum_{n=0}^{y_i-1} p_{n,i} \geqq \frac{a}{1-a} \sum_{j=0}^{M_i-1} \left(\frac{M_i-j}{M_i}\right) p_{j+y_i} .$$

From (II-1), $p_{n,i}$ is proportional to $\overline{\lambda}_i^n$ for all cases. Therefore, the following notation is introduced:

$$p_{n,i} = Q_{n,i}\overline{\lambda}_i^n ,$$

where $Q_{n,i}$ is a function of M_i , μ_i , c_i , and y_i . In that the time period remains the same for the duration of this discussion, the subscript i will be dropped to ease the reading. The availability constraint can be rewritten as

$$\frac{1-a}{a} \geq \frac{\sum_{j=0}^{M-1} (1-\frac{j}{M}) Q_{j+y} \bar{\lambda}^{y+j}}{\sum_{n=0}^{y-1} Q_n \bar{\lambda}^n} .$$

Assume that the availability constraint is satisfied for some c',y' . We will now show that the pair (c',y') also satisfies

$$\frac{1-a}{a} \geq \frac{\sum_{j=0}^{M} (1-\frac{j}{M}) Q_{j+y} \hat{\lambda}^{y+j}}{\sum_{n=0}^{y-1} Q_n \hat{\lambda}^n} ,$$

provided that $\hat{\lambda} < \bar{\lambda}$. Equivalently, it will be shown that

$$\frac{\sum_{j=0}^{M} (1-\frac{j}{M}) Q_{j+y} \bar{\lambda}^{y+j}}{\sum_{n=0}^{y-1} Q_n \bar{\lambda}^n} \geq \frac{\sum_{j=0}^{M} (1-\frac{j}{M}) Q_{j+y} \hat{\lambda}^{y+j}}{\sum_{n=0}^{y-1} Q_n \hat{\lambda}^n}$$

or

$$\left\{\sum_{j=0}^{M} (1-\frac{j}{M}) Q_{j+y} \bar{\lambda}^{y+j}\right\} \left\{\sum_{n=0}^{y-1} Q_n \hat{\lambda}^n\right\} \geq \left\{\sum_{j=0}^{M} (1-\frac{j}{M}) Q_{j+y} \hat{\lambda}^{y+j}\right\} \left\{\sum_{n=0}^{y-1} Q_n \bar{\lambda}^n\right\}$$

$$\sum_{j=0}^{M} \sum_{n=0}^{y-1} (1-\frac{j}{M}) Q_{j+y} Q_n \bar{\lambda}^{y+j} \hat{\lambda}^n - \sum_{j=0}^{M} \sum_{n=0}^{y-1} (1-\frac{j}{M}) Q_{j+y} Q_n \hat{\lambda}^{y+j} \bar{\lambda}^n \geq 0$$

$$\sum_{j=0}^{M} \sum_{n=0}^{y-1} (1-\frac{j}{M}) Q_{j+y} Q_n \hat{\lambda}^{y+j+n} \left(\frac{\bar{\lambda}^{y+j}}{\hat{\lambda}^{y+j}} - \frac{\bar{\lambda}^n}{\hat{\lambda}^n}\right) \geq 0 .$$

but for $\bar{\lambda}/\hat{\lambda} \geq 1$ and $y+j > n$,

$$\left(\frac{\bar{\lambda}}{\hat{\lambda}}\right)^{y+j} - \left(\frac{\bar{\lambda}}{\hat{\lambda}}\right)^n \geq 0 .$$

Since all other terms in the summation are greater than 0, this inequality holds. Therefore, if the inequality associated with the availability constraint holds for a pair (c',y') and $\bar{\lambda}$, then it also holds for the same pair (c',y') when $\hat{\lambda} < \bar{\lambda}$. This implies that for any time period, any point that satisfies the availability constraint using the actual $\bar{\lambda}_i$ will also satisfy the availability constraint using $\hat{\lambda}_i < \bar{\lambda}_i$; so if the extended problem is associated with using a suitable set of $\hat{\lambda}_i$ we have $S \subseteq T$. Here S is the feasible region of the original problem, using $\bar{\lambda}_i$ for evaluating the availability constraints, and T is the feasible region associated with the extended problem, using $\hat{\lambda}_i \leqq \bar{\lambda}_i$ for evaluating the availability constraints.

The problem remains to determine a suitable set of $\hat{\lambda}_i$ values. From Equation (II-2) we see that $\bar{\lambda}_1$ and $\bar{\lambda}_2$ are independent of any policy and are dependent only on the specified parameters. Thus,

$$\bar{\lambda}_1 = \lambda_1$$

$$\bar{\lambda}_2 = \begin{cases} \lambda_2 + (M_1/M_2)(\lambda_1 - \lambda_2) , & M_2 \geqq M_1 \\ \lambda_1 , & M_2 < M_1 \end{cases} .$$

For subsequent time periods $\bar{\lambda}_i$ is specified by:

$$\bar{\lambda}_i = \begin{cases} \lambda_i + (M_{i-1}/M_i)(\bar{\lambda}_{i-1} - \lambda_i) + (\bar{R}_{i-1}/M_i)(\lambda_{i-1} - \bar{\lambda}_{i-1}) , & M_i \geqq M_{i-1} \\ \bar{\lambda}_{i-1} + (\bar{R}_{i-1}/M_{i-1})(\lambda_{i-1} - \bar{\lambda}_{i-1}) , & M_i < M_{i-1} \end{cases} ,$$

$$\bar{R}_i = \bar{\lambda}_i \left[M_i - \sum_{n=y_i+1}^{M_i+y_i} (n-y_i) p_{n,i} \right] .$$

Introducing the notation

$$X_i = \sum_{n=y_i+1}^{M_i+y_i} (n-y_i) p_{n,i} ,$$

we get

$$\bar{\lambda}_i = \begin{cases} \lambda_i + (M_{i-1}/M_i)(\bar{\lambda}_{i-1} - \lambda_i) + \bar{\lambda}_{i-1}(\lambda_{i-1} - \bar{\lambda}_{i-1})(M_{i-1} - X_{i-1}) \,/\, M_i \,, & M_i \geq M_{i-1} \\ \bar{\lambda}_{i-1} + \lambda_{i-1}(\lambda_{i-1} - \bar{\lambda}_{i-1})(M_{i-1} - X_{i-1}) \,/\, M_{i-1} \,, & M_i < M_{i-1} \end{cases} . \qquad \text{(III-1)}$$

We now have $\bar{\lambda}_i$ totally dependent on the choice of the policy of the previous time periods contained in the X_{i-1} . Note that $\bar{\lambda}_i$ is not dependent on X_i . The bounds on $\bar{\lambda}_i$ can be determined.

Limits on X_i can be determined by noting that, since $p_{n,i}$ is a probability, it follows that $0 \leq p_{n,i} \leq 1$ and $\sum_{n=0}^{M_i+y_i} p_{n,i} = 1$. Also, in the summation defining X_i , we have $y_i+1 \leq n \leq M_i+y_i$. Therefore, for $X_i = \sum_{n=y_i+1}^{M_i+y_i} (n-y_i)p_{n,i}$,

$$0 \leq X_i \leq M_i \sum_{n=y_i+1}^{M_i+y_i} p_{n,i}$$

or

$$0 \leq X_i \leq M_i \ .$$

To determine the initial extended problem in the branch and bound procedure, we must find a $\hat{\lambda}_i \leq \bar{\lambda}_i$; therefore, we will choose a value of X_{i-1} such that $\hat{\lambda}_i = \min_{X_{i-1}} (\bar{\lambda}_i)$. Since $0 \leq X_{i-1} \leq M_{i-1}$, from (III-1) the rule for choosing $\bar{\lambda}_i$ should be determined either by setting

$$X_{i-1} = 0 \ , \qquad \lambda_{i-1} \leq \bar{\lambda}_{i-1}$$

or

$$X_{i-1} = M_{i-1} \ , \qquad \lambda_{i-1} > \bar{\lambda}_{i-1} \ .$$

However, the problem is more complex than that. All subsequent $\overline{\lambda}_j$'s , j greater than i , are dependent on the value of X_i . While X_i may be chosen to minimize $\overline{\lambda}_{i+1}$, it may maximize $\overline{\lambda}_{i+2}$, and so on.

By using the explicit dependence of $\overline{\lambda}_i$ on $\overline{\lambda}_{i-1}$ we can specify a condition on λ_i to ensure that the minimization of $\overline{\lambda}_i$ will also minimize $\overline{\lambda}_j$, for j greater than i . Since λ_i is a parameter of the problem, if, when the problem is specified, the λ_i's do not satisfy this bound, there is no assurance that the procedures described herein will work.

First note, from expression (II-2), that $\overline{\lambda}_i$ is an average, and that, therefore, $\overline{\lambda}_i \leqq \max\limits_{j \leqq i} (\lambda_j)$. Next we examine the behavior of $\overline{\lambda}_i$. If $\overline{\lambda}_i$ is a monotonically increasing function of $\overline{\lambda}_{i-1}$, then we know that the minimum value of $\overline{\lambda}_i$ will occur at the minimum value of $\overline{\lambda}_{i-1}$. Examining (III-1), we see that $\overline{\lambda}_i$ is quadratic in $\overline{\lambda}_{i-1}$ if $X_{i-1} < M_{i-1}$, and linear in $\overline{\lambda}_{i-1}$ if $X_{i-1} = M_{i-1}$. Therefore, for $X_{i-1} = M_{i-1}$, since we have monotonicity, the minimum $\overline{\lambda}_i$ is attained when $\overline{\lambda}_{i-1}$ is minimum. For $X_{i-1} < M_{i-1}$, if $\overline{\lambda}_{i-1}$ is constrained to be less than $\overline{\lambda}^*_{i-1}$, where $\partial\overline{\lambda}_i(\overline{\lambda}^*_{i-1})/\partial\overline{\lambda}^*_{i-1} = 0$, then the minimum $\overline{\lambda}_i$ is attained when $\overline{\lambda}_{i-1}$ is minimum; i.e., $\overline{\lambda}_i(\overline{\lambda}_{i-1})$ is a monotonically increasing function of $\overline{\lambda}_{i-1}$ for $\overline{\lambda}_{i-1} < \overline{\lambda}^*_{i-1}$.

Evaluating $\partial\overline{\lambda}_i/\partial\overline{\lambda}_{i-1}$ using (III-1),

$$\frac{\partial\overline{\lambda}_i}{\partial\overline{\lambda}_{i-1}} = \begin{cases} \dfrac{M_{i-1}}{M_i} + \dfrac{M_{i-1} - X_{i-1}}{M_i}(\lambda_{i-1} - 2\overline{\lambda}_{i-1}) , & M_i \geqq M_{i-1} \\ 1 + \dfrac{M_{i-1} - X_{i-1}}{M_{i-1}}(\lambda_{i-1} - 2\overline{\lambda}_{i-1}) , & M_i < M_{i-1} \end{cases} .$$

Setting $\partial\overline{\lambda}_i/\partial\overline{\lambda}_{i-1}$ to zero to determine $\overline{\lambda}^*_{i-1}$, we get

$$\overline{\lambda}^*_{i-1} = \frac{M_{i-1}}{2(M_{i-1} - X_{i-1})} + \frac{\lambda_{i-1}}{2} , \quad X_i < M_{i-1} .$$

Since

$$0 \leqq X_{i-1} < M_{i-1} ,$$

then

$$\overline{\lambda}^*_{i-1} \geqq \frac{1}{2} + \frac{\lambda_{i-1}}{2} .$$

Therefore, if

$$\lambda_i < \frac{1}{2} , \text{ for all } i , \text{ since } \overline{\lambda}_i \leqq \lambda_i ,$$

then we know that $\overline{\lambda}_i < \frac{1}{2}$ for all i . Therefore $\overline{\lambda}_{i-1}$ can never be greater than $\overline{\lambda}^*_{i-1}$, so $\overline{\lambda}_i$ is a monotonically increasing function of $\overline{\lambda}_{i-1}$ and the minimum $\overline{\lambda}_i$ will be attained for the minimum $\overline{\lambda}_{i-1}$.

This is a sufficiency condition. Tighter requirements could be found for the upper bound of $\overline{\lambda}_{i-1}$ to assure that the minimum value of $\overline{\lambda}_i$ is attained at the minimum value of $\overline{\lambda}_{i-1}$. In that the expression (II-2) is an approximation to the actual failure rate of the process, and the approximation is good only for small values of λ_i , this sufficiency condition does not impose any additional practical restrictions on the problem. In the next chapter we will see that the numerical data more than satisfy this condition.

It is now possible to specify the branch and bound procedure. The original problem is

$$\underset{c_i, y_i, \forall i}{\text{Min}} \quad \sum_{i=1}^{k} d^i \left[C_{1,i} c_i + C_{2,i} y_i \right]$$

$$\text{s.t.} \quad \sum_{n=0}^{y_i - 1} q_{n,i} \geqq 0.90 \,, \quad \forall i$$

$$c_i \geqq c_{i-1} \,, \qquad \forall i$$

$$y_i \geqq y_{i-1} \,, \qquad \forall i$$

$$y_1 \geqq 0$$

$$c_1 \geqq 0$$

c_i and y_i are integers for all i .

The $q_{n,i}$ are implicit functions of $\overline{\lambda}_i$ which are themselves implicit functions of c_j, y_j for j less than i . The extended problem is created by replacing $\overline{\lambda}_i$ by $\hat{\lambda}_i$. The $\hat{\lambda}_i$ are independent of c_j, y_j and $\hat{\lambda}_i \leqq \overline{\lambda}_i$. The $\hat{\lambda}_i$ are given by:

for $\hat{\lambda}_{i-1} > \lambda_{i-1}$, (III-2a)

$$\hat{\lambda}_i = \begin{cases} \lambda_i + (M_{i-1}/M_i)(\hat{\lambda}_{i-1} - \lambda_i) + \hat{\lambda}_{i-1}(M_{i-1}/M_i)(\lambda_{i-1} - \hat{\lambda}_{i-1}) \,, & M_i \geqq M_{i-1} \\ \hat{\lambda}_{i-1} + \hat{\lambda}_{i-1}(\lambda_{i-1} - \hat{\lambda}_{i-1}) \,, & M_i < M_{i-1} \end{cases}$$

or for $\hat{\lambda}_{i-1} < \lambda_{i-1}$,

$$\hat{\lambda}_i = \begin{cases} \lambda_i + (M_{i-1}/M_i)(\hat{\lambda}_{i-1} - \lambda_i) \,, & M_i \geqq M_{i-1} \\ \hat{\lambda}_{i-1} \,, & M_i < M_{i-1} \end{cases} \quad . \qquad \text{(III-2b)}$$

It has been shown in this section that a valid extended problem is creaed by this substitution.

The extended problem is solved by techniques which will be explained in the following sections. If the solution is feasible, the problem is solved. If the solution is not feasible, branching occurs.

If the set $\{(c_j, y_j);\ j < i\}$ is known, it is possible to calculate $\overline{\lambda}_i$ exactly. Therefore, if a set $\{(c_j, y_j);\ j < i\}$ is specified for all time periods, the $\overline{\lambda}_i$ for all time periods can be calculated exactly, and this policy can be tested to determine if it is feasible.

The branch and bound technique when applied to this problem proceeds in the following manner. The first subproblem is generated by calculating $\hat{\lambda}_i$ for all time periods. For time periods 1 and 2, $\hat{\lambda}_i = \overline{\lambda}_i$, since for these time periods, $\overline{\lambda}_i$ is dependent only on the input parameters and is, therefore, known exactly. For all other time periods $\hat{\lambda}_i$ is calculated using Equations (III-2a) or (III-2b). This subproblem is solved using the techniques described in the following sections. If the solution to this subproblem is feasible, we have found the solution to the original problem and we are finished. If the solution is infeasible, branching is performed.

In solving the first subproblem, a set of m pairs $\{(c_2, y_2)_\ell;\ \ell \leq m\}$ is determined for the second time period. We know that the solution to the initial problem must contain one of these pairs as the optimal pair for the second time period. This is because, in the procedure that is described in the following sections, for a given $\overline{\lambda}_i$ all of the extreme points of the region satisfying the constraints are enumerated. This enumeration is dependent on $\hat{\lambda}_i$, but for the second time period (as well as for the first time period) $\hat{\lambda}_2 = \overline{\lambda}_2$.

If there are m such pairs, then for each one of these pairs $\{(c_2,y_2)_\ell;\ \ell \leq m\}$, a $\bar{\lambda}_3^\ell$ can be calculated. If the optimal policy associated with the solution of the initial problem contains $(c_2,y_2)_\ell$, then $\bar{\lambda}_3^\ell$ is the exact $\bar{\lambda}_3$ for the third time period. Thus we have created m new subproblems, each having a $\bar{\lambda}_3^\ell$ associated with each of the m pairs $(c_2,y_2)_\ell$. For all subsequent time periods (≥ 4) for each subproblem, the $\hat{\lambda}_i$ are calculated using (III-2a) or (III-2b).

Each new subproblem is solved using the solution techniques. The problem that has the best lower bound is then tested for feasibility. If it is feasible, the solution has been found. If it is not feasible, branching continues based upon the enumerated set of pairs $(c_3,y_3)_\ell$. If the solution to the initial problem contains the (c,y) pair of the second time period that generated the subproblem from which the current branching is to occur, then the optimal policy for the third time period for the solution of the initial problem must come from the set of pairs $(c_3,y_3)_\ell$.

The new branches are generated in the same manner previously performed, with an exact $\bar{\lambda}_4^\ell$ calculated for each of the pairs $(c_3,y_3)_\ell$. There is a new subproblem associated with each of the $\bar{\lambda}_4^\ell$, with the $\hat{\lambda}_i$ calculated for subsequent time periods (≥ 5) using (III-2a) and (III-2b). After the branching a new best lower bound is determined and tested for feasibility.

This procedure continues until a feasible best lower bound is found. Branching is always based upon the set of pairs $(c_i,y_i)_\ell$ for the last time period in the subproblem associated with the best lower bound, for which the exact $\bar{\lambda}_i$ has been used. The determination of an exact $\overline{\lambda}_i$ is dependent on the specification of (c,y) pairs for all previous time periods.

This procedure is equivalent to taking all possible combinations of policies for each time period based upon the enumeration of those points that satisfy the constraints. Since it is not possible to enumerate correctly the set of policies in a given time period until the policies of all the previous time periods have been specified ($\overline{\lambda}_i$ cannot be calculated), it is necessary to do this enumeration in a sequential order. The advantage of the branch and bound procedure is that it offers the hope that the solution to the initial problem will be found before it is necessary to go through all of the branches. This hope has been justified in practice.

Since, for a given $\overline{\lambda}_i$, we are interested in only a finite number of pairs (c_i,y_i) that satisfy the constraints of the problem, as will be discussed in the next section, and since there is a finite number of time periods, there can only be a finite number of branches. Therefore, the branch and bound procedure must converge [8].

The procedure can now be summarized as follows:

(1) Create the extended problem by replacing $\overline{\lambda}_i$ by the approximation $\hat{\lambda}_i$, as given in Equations (III-2a) and (III-2b).

(2) Solve the extended problem.

(3) If the solution is feasible, the problem is solved.

(4) If the solution is infeasible, branch.

(5) Based upon the last time period ℓ for which $\hat{\lambda}_i = \overline{\lambda}_k$ for all k less than ℓ ,

calculate the set $\overline{\lambda}^{i}_{\ell}$. Create a new set of problems.

(6) Solve the new set of problems.

(7) If the problem associated with the BLB is feasible, it is the exact solution and the problem is complete.

(8) If the problem is infeasible, branch as in Step 5.

We will next look at the techniques that are required to solve the extended problem.

III.4 Enumeration Techniques

The objective function has been manipulated into a linear form; therefore, if there is a bounded solution then there is one which will be on the convex hull of the feasible region. For the branch and bound approach that is being used, it is also true that for any stage of the procedure the solution will be found on the convex hull of the feasible region of a subproblem. A method of generating the convex hull is therefore required. The enumeration technique is the first step of that process.

For each time period, and for each subproblem of the branch and bound method, the convex hull of the region satisfying the availability constraint can be enumerated. In this section it will be shown that the enumeration is finite. Also presented in this section is the algorithm used in the enumeration. Because the following discussion is independent of time, the subscript i will again be dropped to simplify notation.

Figure III-1 illustrates the procedure used to enumerate the region satisfying the availability constraint. There are three

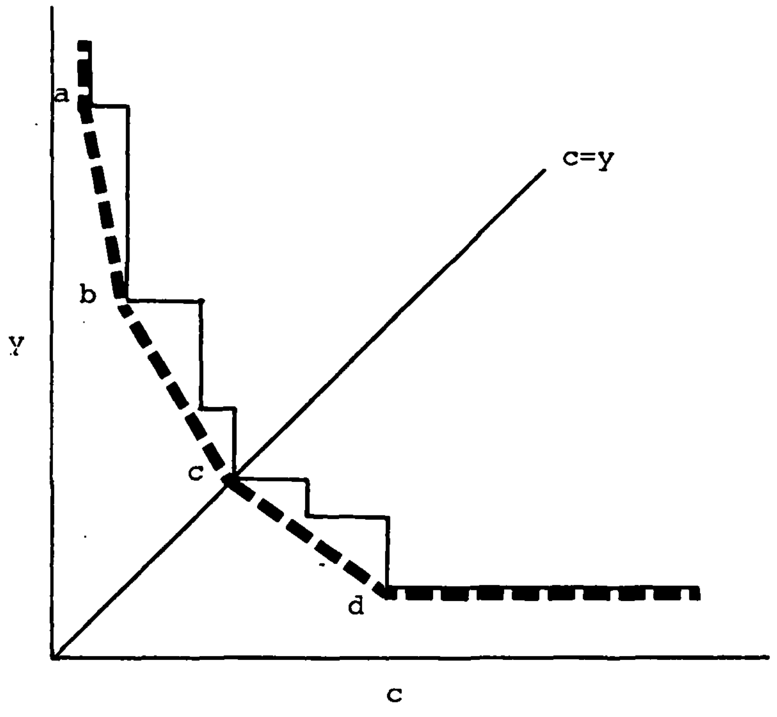


Fig. III-1. Boundary of region satisfying availability constraint.

lines of interest on this figure: the solid step-like line, the dashed line, and the diagonal line $c=y$. Each of these is significant in the discussion of the enumeration process.

Because of the physical reality of the problem (fractional spares or repair channels are meaningless) and the nature of the equation defining the state of the system, one of the constraints of the problem is that both c, the number of repair channels, and y, the number of spares, are integer. Therefore, in Figure III-1 we are interested only in the lattice of points associated with the integer pairs (c,y). The solid step-like line is the boundary of the region for which the pair (c,y) satisfies the availability constraint. As will be proven, all integer pairs to the right and above this line satisfy the availability constraint. To the left or below this line, the constraint is not satisfied. For a given y, the minimum c for which the availability constraint is satisfied is given by a point on this line. If a c is less than the leftmost value of the line, the availability constraint cannot be satisfied for that c, and for a y less than the lowest value of the line, the availability constraint cannot be satisfied for that y.

The dashed line is the boundary of the convex hull of the region satisfying the availability constraint. Each of the vertices of this convex hull must be at one of the (c,y) pairs for the boundary of the region satisfying the availability constraint. Not all of the (c,y) pairs that lie on the boundary of the region satisfying the availability constraint will lie on its convex hull.

The solid diagonal line $c=y$ is important in this enumeration technique in that the equations specifying the state of the system depend on which side of the line the state falls. Equations (II-1), *a*, *b*, and *c* hold when $c \leq y$ (to the left of the line), and Equations (II-1), *d*, *e*, and *f* hold when $c \geq y$ (to the right of the line).

The enumeration procedure starts by finding the smallest c for which the availability constraint is satisfied, and determines the minimum y associated with that c for which the availability constraint is satisfied (point a). Then, as will be shown, since all points above and to the right of that point (a) satisfy the availability constraint, we search for the next point to the right and below for which the availability constraint is satisfied. The procedure goes from point to point in this manner. If we are searching for a point to the left of the $c=y$ line, Equations (II-1), *a*, *b*, and *c* are used. As soon as $c \geqq y$, Equations (II-1), *d*, *e*, and *f* are used. The procedure continues by incrementing c or decrementing y until the availability constraint cannot be satisfied either for $y=1$ or for $c = y+M$.

It is proven in this section that the hull can be generated in this manner in a finite number of steps. The outline for the algorithm used in the computer program is also presented.

The expression given in Appendix B, (B-1), for the availability constraint will be used as we examine the nature of the region satisfying the availability constraint and generate expressions used in the computer algorithm.

By the nature of the problem, one intuitively feels that if the availability constraint is satisfied for a (c,y) pair, it should also be satisfied for (c',y') when $c' \geqq c$ and $y' \geqq y$. This shall now be proven.

Theorem 2: If the pair (c,y) satisfies the availability constraint, then any pair (c',y'), where $c' \geqq c$ and $y' \geqq y$, satisfies the availability constraint.

Proof: *Part I,* $c \leqq y$.

The state probabilities are given by Equation (II-1), *a*, *b*, and *c*, which are repeated here.

$$p_n = \begin{cases} \dfrac{M^n}{n!}\left(\dfrac{\bar{\lambda}}{\mu}\right)^n p_0 \, , & (0 \leqq n \leqq c) \qquad a \\[2ex] \dfrac{M^n}{c^{n-c}\; c!}\left(\dfrac{\bar{\lambda}}{\mu}\right)^n p_0 \, , & (c \leqq n \leqq y) \qquad b \\[2ex] \dfrac{M^y M!}{(M-n+y)!\; c^{n-c}\; c!}\left(\dfrac{\bar{\lambda}}{\mu}\right)^n p_0 \, , & (y \leqq n \leqq y+M) \qquad c \, . \end{cases} \qquad \text{(II-1)}$$

Using Equation (B-1) from Appendix B, the availability constraint for $c \leqq y$ can be written:

$$\sum_{n=0}^{c-1} \frac{\rho^n}{n!} + \sum_{n=c}^{y-1} \frac{\rho^n}{c^{n-c}\; c!} \geqq \left(\frac{a}{a-1}\right)\rho^y \sum_{j=0}^{M-1} \frac{M!\,(M-j)}{M(M-j)!} \frac{1}{c^{j+y-c}\; c!} \left(\frac{\rho}{M}\right)^j ,$$

where $\rho = \dfrac{M\bar{\lambda}}{\mu}$, and p_0 has been eliminated from all terms. This can be rewritten as

$$\sum_{n=0}^{c-1} \frac{\rho^n}{n!} + \sum_{n=c}^{y-1} \frac{\rho^n}{c^{n-c}\; c!} \geqq \left(\frac{a}{1-a}\right)\rho^y \sum_{j=0}^{M-1} \frac{(M-1)!}{(M-1-j)!} \frac{1}{c^{j+y-c}\; c!} \left(\frac{\rho}{M}\right)^j . \qquad \text{(III-3)}$$

Part I(a).

It will now be proven that if (III-3) is satisfied for a (c,y) pair then it will be satisfied for a (c',y) pair, where $c' \geqq c$ and $c' \leqq y$. (It will subsequently be shown that the availability constraint remains satisfied for $c' > y$.) For $c' = c+1$, the availability constraint, as expressed by (III-3), is

$$\sum_{n=0}^{c} \frac{\rho^n}{n!} + \sum_{n=c+1}^{y-1} \frac{\rho^n}{(c+1)^{n-c-1}\; (c+1)!} \geqq$$

$$\left(\frac{a}{1-a}\right)\rho^y \sum_{j=0}^{M-1} \frac{(M-1)!}{(M-1-j)!} \frac{1}{(c+1)^{j+y-c-1}\; (c+1)!} \left(\frac{\rho}{M}\right)^j ,$$

or

$$\sum_{n=0}^{c-1} \frac{\rho^n}{n!} + \frac{\rho^c}{c!} - \frac{\rho^c}{c!} + \sum_{n=c}^{y-1} \frac{\rho^n}{(c+1)^{n-c}\ c!} \geqq$$

$$\left(\frac{a}{1-a}\right)\rho^y \sum_{j=0}^{M-1} \frac{(M-1)!}{(M-1-j)!} \frac{1}{(c+1)^{j+y-c}\ c!} \left(\frac{\rho}{M}\right)^j .$$

Multiplying by $[(c+1)/c]^{y-c}$,

$$\left(\frac{c+1}{c}\right)^{y-c} \sum_{n=0}^{c-1} \frac{\rho^n}{n!} + \sum_{n=c}^{y-1} \frac{\rho^n}{c!\ c^{n-c}} \left(\frac{c+1}{c}\right)^{y-n} \geqq$$

$$\left(\frac{a}{1-a}\right)\rho^y \sum_{j=0}^{M-1} \frac{(M-1)!}{(M-1-j)!} \frac{1}{c^{j+y-c}\ c!} \left(\frac{\rho}{M}\right)^j \left(\frac{c}{c+1}\right)^j . \tag{III-4}$$

Since for $y \geqq c$, $[(c+1)/c]^{y-c} \geqq 1$, and for $n \leqq y-1$, $[(c+1)/c]^{y-n} \geqq 1$, a comparison of the left-hand terms of (III-4) to the left-hand terms of (III-3) shows that both of the terms on the left-hand side of (III-4) are greater than the equivalent terms in (III-3). Also, since for $j \geqq 0$, $[c/(c+1)]^j \leqq 1$, the right-hand side of (III-4) is less than the right-hand side of (III-3). Therefore, if the relation (III-3) is satisfied, then the relation (III-4) must be satisfied. So if the availability constraint as given in (III-3) is satisfied for a (c,y) pair, then it must also be satisfied for $(c+1,y)$, and consequently for (c',y) when $c' \geqq c$.

Part I(b).

Still assuming that $c \leqq y$, it will now be proven that if (III-3) is satisfied for a (c,y) pair then it will be satisfied for (c,y') when $y' \geqq y$.

Let $y' = y+1$. The availability constraint can then be written

$$\sum_{n=0}^{c-1} \frac{\rho^n}{n!} + \sum_{n=c}^{y} \frac{\rho^n}{c^{n-c}\ c!} \geqq \left(\frac{a}{1-a}\right)\rho^{y+1} \sum_{j=0}^{M-1} \frac{(M-j)}{(M-j)!} \frac{M!}{M} \frac{1}{c^{j+y+1-c}\ c!} \left(\frac{\rho}{M}\right)^j .$$

Rewriting the right-hand side,

$$\sum_{n=0}^{c-1} \frac{\rho^n}{n!} + \sum_{n=c}^{y} \frac{\rho^n}{c^{n-c}\, c!} \geq \left(\frac{a}{1-a}\right) \frac{\rho}{c}\, \rho^y \sum_{j=0}^{M-1} \frac{(M-1)!}{(M-1-j)!} \frac{1}{c^{j+y-c}\, c!} \left(\frac{\rho}{M}\right)^j .$$

The left-hand side can be written as

$$1 + \sum_{n=1}^{c-1} \frac{\rho^n}{n!} + \frac{\rho^c}{c!} + \sum_{n=c+1}^{y} \frac{\rho^n}{c^{n-c}\, c!} .$$

Regrouping terms, this becomes

$$1 + \sum_{n=1}^{c} \frac{\rho^n}{n!} + \sum_{n=c+1}^{y} \frac{\rho^n}{c^{n-c}\, c!} .$$

Factoring out ρ/c , we get

$$\frac{\rho}{c} \left\{ \frac{c}{\rho} + \sum_{n=1}^{c} \frac{\rho^{n-1}}{n!} c + \sum_{n=c+1}^{y} \frac{\rho^{n-1}}{c^{n-1-c}\, c!} \right\} .$$

Redefining the indices gives

$$\frac{\rho}{c} \left\{ \frac{c}{\rho} + \sum_{m=0}^{c-1} \frac{\rho^m}{m!} \frac{c}{m+1} + \sum_{m=c}^{y-1} \frac{\rho^m}{c^{m-c}\, c!} \right\} ;$$

therefore, the inequality becomes

$$\frac{\rho}{c} \left\{ \frac{c}{\rho} + \sum_{n=0}^{c-1} \frac{\rho^n}{n!} \frac{c}{n+1} + \sum_{n=c}^{y-1} \frac{\rho^n}{c^{n-c}\, c!} \right\} \geq$$

$$\left(\frac{a}{1-a}\right) \frac{\rho}{c}\, \rho^y \sum_{j=0}^{M-1} \frac{(M-1)!}{(M-1-j)!} \frac{1}{c^{j+y-c}\, c!} \left(\frac{\rho}{M}\right)^j .$$

Eliminating the ρ/c factor from both sides, we are left with the inequality

$$\frac{c}{\rho} + \sum_{n=0}^{c-1} \frac{\rho^n}{n!} \frac{c}{n+1} + \sum_{n=c}^{y-1} \frac{\rho^n}{c^{n-c}\, c!} \geq \left(\frac{a}{1-a}\right) \rho^y \sum_{j=0}^{M-1} \frac{(M-1)!}{(M-1-j)!} \frac{1}{c^{j+y-c}\, c!} \left(\frac{\rho}{M}\right)^j . \tag{III-5}$$

Since for $n \leq c-1$, $[c/(n+1)] \geq 1$, the left-hand side of (III-5) is greater than the left-hand side of (III-3). Moreover, the right-hand sides of (III-3) and (III-5) are identical. Therefore, if (III-3) is satisfied for a (c,y) pair then (III-5) must be satisfied for (c,y') when $y' = y+1$, and consequently for any pair (c,y') when $y' \geqq y$.

In Parts I(a) and I(b) of this proof we have shown that when $c \leqq y$, if a pair (c,y) satisfies the availability constraint then (c',y) and (c,y') satisfy the availability constraint, respectively when $c' \geqq c$ and $c' \leqq y$, and when $y' \geqq y$. This implies that any pair (c',y'), $c' \geqq c$ and $y' \geqq y$, will also satisfy the availability constraint when $c' \leqq y$. This concludes Part I of this proof.

Part II, $c \geqq y$.

For $c \geqq y$ the state probabilities are given by Equation (II-1), *d*, *e*, and *f*, which is repeated here.

$$p_n = \begin{cases} \dfrac{M^n}{n!}\left(\dfrac{\bar{\lambda}}{\mu}\right)^n p_0, & (0 \leqq n \leqq y) \quad d \\[2ex] \dfrac{M^y M!}{(M-n+y)!\; n!}\left(\dfrac{\bar{\lambda}}{\mu}\right)^n p_0, & (y \leqq n \leqq c) \quad e \\[2ex] \dfrac{M^y M!}{(M-n+y)!\; c^{n-c}\; c!}\left(\dfrac{\bar{\lambda}}{\mu}\right)^n p_0, & (c \leqq n \leqq y+M) \quad f \end{cases} \qquad \text{(II-1)}$$

Using Equation (B-1) the availability constraint for $c \geqq y$ can be written as:

$$\sum_{n=0}^{y-1} \frac{\rho^n}{n!} \geqq \left(\frac{a}{1-a}\right)\rho^y \left\{ \sum_{j=0}^{c-y-1} \frac{(M-j)\; M!}{(M-j)!\; M(y+j)!}\left(\frac{\rho}{M}\right)^j + \sum_{j=c-y}^{M-1} \frac{(M-j)\; M!}{(M-j)!\; Mc^{j+y-c}\; c!}\left(\frac{\rho}{M}\right)^j \right\},$$

where, again, $\rho = \dfrac{M\bar{\lambda}}{\mu}$ and p_0 has been eliminated from all terms. This can be rewritten as:

$$\sum_{n=0}^{y-1} \frac{\rho^n}{n!} \geqq \left(\frac{a}{1-a}\right)\rho^y \left\{ \sum_{j=0}^{c-y-1} \frac{(M-1)!}{(M-1-j)!\,(y+j)!} \left(\frac{\rho}{M}\right)^j + \frac{c^c}{c!} \sum_{j=c-y}^{M-1} \frac{(M-1)!}{(M-1-j)!} \left(\frac{\rho}{M}\right)^j c^{-(j+y)} \right\} . \tag{III-6}$$

Part II(a).

It will now be proven that if (III-6) is satisfied for a (c,y) pair then it will be satisfied for a (c',y) pair where $c' \geqq c$. For $c' = c+1$, (III-6) becomes

$$\sum_{n=0}^{y-1} \frac{\rho^n}{n!} \geqq \left(\frac{a}{1-a}\right)\rho^y \left\{ \sum_{j=0}^{c-y} \frac{(M-1)!}{(M-1-j)!\,(y+j)!} \left(\frac{\rho}{M}\right)^j + \frac{(c+1)^{c+1}}{(c+1)!} \sum_{j=c-y+1}^{M-1} \frac{(M-1)!}{(M-1-j)!} \left(\frac{\rho}{M}\right)^j (c+1)^{-(j+y)} \right\} . \tag{III-7}$$

By changing the limits on the right-hand summation and eliminating a $(c+1)$ from the numerator and denominator of the coefficient of the last summation, we get

$$\sum_{n=0}^{y-1} \frac{\rho^n}{n!} \geqq \left(\frac{a}{1-a}\right)\rho^y \left\{ \sum_{j=0}^{c-y-1} \frac{(M-1)!}{(M-1-j)!\,(y+j)!} \left(\frac{\rho}{M}\right)^j + \frac{(M-1)!}{(M-1-c+y)!\,c!} \left(\frac{\rho}{M}\right)^{c-y} - \frac{(M-1)!}{(M-1-c+y)!\,c!} \left(\frac{\rho}{M}\right)^{c-y} + \frac{(c+1)^c}{c!} \sum_{j=c-y}^{M-1} \frac{(M-1)!}{(M-1-j)!} \left(\frac{\rho}{M}\right)^j (c+1)^{-(j+y)} \right\} ,$$

or

$$\sum_{n=0}^{y-1} \frac{\rho^n}{n!} \geqq \left(\frac{a}{1-a}\right)\rho^y \left\{ \sum_{j=0}^{c-y-1} \frac{(M-1)!}{(M-1-j)!\,(y+j)!} \left(\frac{\rho}{M}\right)^j + \frac{(c+1)^c}{c!} \sum_{j=c-y}^{M-1} \frac{(M-1)!}{(M-1-j)!} \left(\frac{\rho}{M}\right)^j (c+1)^{-(j+y)} \right\} . \tag{III-8}$$

Comparing (III-8) and (III-6) we see that all terms are identical except for the last summation on the right. Each term in the last summation of (III-8) differs by a factor of $[c/(c+1)]^{j+y-c}$ from the comparable term in (III-6).

Since in the last summation $j \geqq c-y$ and $c/(c+1) < 1$, it is obvious that the last summation in (III-8) is less than the last summation in (III-6). Therefore, if the inequality (III-6) is satisfied, then the inequality (III-7) is satisfied. So for $c \geqq y$, if the availability constraint is satisfied for (c,y) then the constraint is satisfied for (c',y) when $c' = c+1$ and, therefore, when $c' \geqq c$.

Part II(b).

It will now be proven that if the availability constraint is satisfied for a pair (c,y) then it will be satisfied for a pair (c,y') when $y' \geqq y$. This was shown to be true when $c = y$ in Part I(b). Now we shall assume that $y < c$ and will show that (III-6) remains satisfied provided that $y' \leqq c$.

Let $y' = y+1$. The availability constraint can then be written

$$\sum_{n=0}^{y'-1} \frac{\rho^n}{n!} \geqq \left(\frac{a}{1-a}\right)\rho^{y'}\left\{\sum_{j=0}^{c-y'-1} \frac{(M-j)}{(M-j)!}\,\frac{M!}{M(y'+j)!}\left(\frac{\rho}{M}\right)^j + \sum_{j=c-y'}^{M-1} \frac{(M-j)}{(M-j)!}\,\frac{M!}{M(c^{j+y'-c})\;c!}\left(\frac{\rho}{M}\right)^j\right\}. \qquad \text{(III-9)}$$

Substituting $y' = y+1$ and manipulating, this can be written

$$\sum_{n=0}^{y} \frac{\rho^n}{n!} \geqq \left(\frac{a}{1-a}\right)\rho^{y+1}\left\{\sum_{j=0}^{c-y-2} \frac{(M-1)!}{(M-1-j)!\;(y+1+j)!}\left(\frac{\rho}{M}\right)^j + \frac{c^c}{c!}\sum_{j=c-y-1}^{M-1} \frac{(M-1)!}{(M-1-j)!}\left(\frac{\rho}{M}\right)^j c^{-(j+y+1)}\right\}.$$

By pulling terms out of the summations and dividing both sides by ρ , the inequality can be rewritten as

$$\frac{1}{\rho} + \sum_{m=0}^{y-1} \frac{\rho^m}{(m+1)!} \geqq \left(\frac{a}{1-a}\right)\rho^y \left\{ \sum_{j=0}^{c-y-1} \frac{(M-1)!}{(M-1-j)!\,(y+1+j)!} \left(\frac{\rho}{M}\right)^j \right.$$

$$- \frac{(M-1)!}{(M+y-c)!\,c!} \left(\frac{\rho}{M}\right)^{c-y-1}$$

$$+ \frac{c^c}{c!} \frac{(M-1)!}{(M+y-c)!} \left(\frac{\rho}{M}\right)^{c-y-1} c^{-c}$$

$$\left. + \frac{c^c}{c!} \sum_{j=c-y}^{M-1} \frac{(M-1)!}{(M-1-j)!} \left(\frac{\rho}{M}\right)^j \frac{c^{-(j+y)}}{c} \right\} .$$

Multiplying by y we get

$$\frac{y}{\rho} + \sum_{n=0}^{y-1} \frac{\rho^n}{n!} \frac{y}{n+1} \geqq \left(\frac{a}{1-a}\right)\rho^y \left\{ \sum_{j=0}^{c-y-1} \frac{(M-1)!}{(M-1-j)!\,(y+j)!} \frac{y}{y+1+j} \left(\frac{\rho}{M}\right)^j \right.$$

$$\left. + \frac{c^c}{c!} \sum_{j=c-y}^{M-1} \frac{(M-1)!}{(M-1-j)!} \left(\frac{\rho}{M}\right)^j c^{-(y+j)} \frac{y}{c} \right\} . \qquad \text{(III-10)}$$

A term by term comparison of (III-10) and (III-6) shows that if the inequality (III-6) is satisfied then (III-10) is also satisfied. On the left-hand side of (III-10) there is an additional term which is greater than zero and since, in the left-hand summation, $n \leqq y-1$, each term of the left-hand summation of (III-10) is greater than the corresponding term in the left-hand summation of (III-6). In the first summation on the right-hand side, since $j \geqq 0$, each term of (III-10) is smaller than its corresponding term in (III-6), and in the second summation on the right-hand side, since $y \leqq c$, each term of (III-10) is also smaller than or equal to the corresponding term of (III-6).

Therefore, if for a pair (c,y) (III-6) is satisfied, then for a pair $(c,y+1)$ the inequality is also satisfied and the same is consequently true for (c,y') when $y' \geqq y$. In Parts II(a) and II(b) of this proof we have shown that, for $y \leqq c$, if a pair (c,y) satisfies the availability constraint then a pair (c',y') also satisfies the constraint when $c' \geqq c$ and $y' \geqq y$.

By Parts I and II we have proven that for $y \geqq c$ or $y \leqq c$ if the availability constraint is satisfied for a pair (c,y) it is also satisfied for (c',y') when $c' \geqq c$ and $y' \geqq y$. //

Referring back to Figure III-1, the geometric interpretation is that any point above and to the right of a boundary point satisfies the availability constraint. This property will be used later in the generation of the feasible region.

For the algorithmic development there is a more convenient form for (III-3). Note that the second summation is a geometric sum. Therefore, for $\rho \neq c$, (III-3) can be written

$$\sum_{n=0}^{c-1} \frac{\rho^n}{n!} + \frac{\rho^c}{c!} \; \frac{1 - \left(\frac{\rho}{c}\right)^{y-c}}{1 - \frac{\rho}{c}} \geqq \left(\frac{a}{1-a}\right) \frac{\rho^c}{c!} \left(\frac{\rho}{c}\right)^{y-c} \sum_{j=0}^{M-1} \left(\frac{\rho}{Mc}\right)^j \frac{(M-1)!}{(M-1-j)!} .$$

The advantage of this form is that the dependence on y has been removed from inside the summation and from its indices. It remains only as an exponent of the term ρ/c.

For convenience, the following notation is introduced:

$$S_1 = \sum_{n=0}^{c-1} \frac{\rho^n}{n!} ,$$

$$S_2 = \sum_{j=0}^{M-1} \left(\frac{\rho}{Mc}\right)^j \frac{(M-1)!}{(M-1-j)!} ,$$

$$A = \frac{a}{1-a} .$$

It is obvious that S_1 and S_2 are both nonnegative and finite. Also, since a is the availability $(0 \leqq a \leqq 1)$, A is nonnegative and finite.

Using this notation, the availability constraint for $y \geqq c$ and $\rho \neq c$ can be written

$$S_1 + \frac{\rho^c}{c!} \frac{1 - \left(\frac{\rho}{c}\right)^{y-c}}{1 - \frac{\rho}{c}} \geqq A \frac{\rho^c}{c!} S_2 \left(\frac{\rho}{c}\right)^{y-c}$$

or

$$S_1 + \frac{\rho^c}{c!} \frac{1}{1 - \frac{\rho}{c}} \geqq \left(AS_2 + \frac{1}{1 - \frac{\rho}{c}}\right) \frac{\rho^c}{c!} \left(\frac{\rho}{c}\right)^{y-c} .$$

Multiplying by $c!/\rho^c$ yields

$$\frac{c!}{\rho^c} S_1 + \frac{1}{1 - \frac{\rho}{c}} \geqq \left(AS_2 + \frac{1}{1 - \frac{\rho}{c}}\right) \left(\frac{\rho}{c}\right)^{y-c} \qquad \text{(III-11)}$$

This form of the availability constraint is very useful in determining the convex hull of the region satisfying the availability constraint.

For $c > \rho$, (III-11) can be written

$$\left(\frac{\rho}{c}\right)^{y-c} \leqq \left(\frac{c!}{\rho^c} S_1 + \frac{1}{1 - \frac{\rho}{c}}\right) \Big/ \left(AS_2 + \frac{1}{1 - \frac{\rho}{c}}\right) .$$

Taking the natural log of both sides,

$$(y-c) \ \ln\left(\frac{\rho}{c}\right) \leqq \ln\left(\frac{c!}{\rho^c} S_1 + \frac{1}{1 - \frac{\rho}{c}}\right) - \ln\left(AS_2 + \frac{1}{1 - \frac{\rho}{c}}\right) .$$

Since $(\rho/c) < 1$, $\ln(\rho/c) < 0$; therefore, dividing by $\ln(\rho/c)$ gives

$$(y-c) \geq \left[\ln\left(\frac{c!}{\rho^c} S_1 + \frac{1}{1 - \frac{\rho}{c}}\right) - \ln\left(AS_2 + \frac{1}{1 - \frac{\rho}{c}}\right)\right] \Big/ \ln\left(\frac{\rho}{c}\right) ,$$

or

$$y > c + \left[\ln\left(\frac{c!}{\rho^c} S_1 + \frac{1}{1 - \frac{\rho}{c}}\right) - \ln\left(AS_2 + \frac{1}{1 - \frac{\rho}{c}}\right)\right] \Big/ \ln\left(\frac{\rho}{c}\right) . \qquad \text{(III-12)}$$

For a specified c, S_1 and S_2 can be evaluated. The boundary of the region satisfying the availability constraint for a given value of c is at the minimum integer value of y that satisfies the availability constraint. Let

$$y^* = c + \left[\ell n \left(\frac{c!}{\rho^c} + S_1 + \frac{1}{1 - \frac{\rho}{c}} \right) - \ell n \left(AS_2 + \frac{1}{1 - \frac{\rho}{c}} \right) \right] / \ell n \left(\frac{\rho}{c} \right) .$$

Then for a given c, a boundary point on the region satisfying the availability constraint is given by the pair $(c, \lceil y^* \rceil)$, where $\lceil a \rceil$ is the smallest integer greater than or equal to a. Remember the relationship (III-12) is only valid for $y \geqq c$ and $c > \rho$.

For y^* to be greater than or equal to c, since $\ell n(\rho/c) < 0$, the following relation must hold:

$$\ell n \left(AS_2 + \frac{1}{1 - \frac{\rho}{c}} \right) \geqq \ell n \left(\frac{c!}{\rho^c} S_1 + \frac{1}{1 - \frac{\rho}{c}} \right) ,$$

or

$$AS_2 \geqq \frac{c!}{\rho^c} S_1 .$$

Therefore, if $(\rho^c/c!)AS_2 \geqq S_1$, we know that for a specified c the point $(c, \lceil y^* \rceil)$ is a boundary point of the feasible region. If, however, $S_1 \geqq (\rho^c/c!)AS_2$, we know from (III-11) that the availability constraint is satisfied for $y = c$ and therefore, either (c,y) is a boundary point when $c = y$, or (c,y'), where $y' < c$, is a boundary point. If (c,y') when $y' < c$ is the boundary point, another method, as described later, is needed to determine that boundary point.

For $c < \rho$ and $y \geqq c$, we can derive another relationship from (III-11). For $c < \rho$, $1/[1 - (\rho/c)] < 0$ and therefore, $AS_2 + 1/[1 - (\rho/c)]$ may or may not be greater than zero and $(c!/\rho^c)S_1 + 1/[1 - (\rho/c)]$ may or may not be greater than zero. However, we can prove that

$$\frac{c!}{\rho^c} S_1 + \frac{1}{1-\frac{\rho}{c}} \leqq 0 .$$

Theorem 3: For $c < \rho$ and $y \geqq c$,

$$\frac{c!}{\rho^c} S_1 + \frac{1}{1-\frac{\rho}{c}} \leqq 0 . \tag{III-13}$$

Proof: The definition of S_1 is

$$S_1 = \begin{cases} \sum_{n=0}^{c-1} \frac{\rho^n}{n!} , & c > 0 \\ 0 , & c = 0 \end{cases} .$$

We can write (III-13) as

$$S_1 \leqq \frac{\rho^c}{c!} \frac{1}{\frac{\rho}{c} - 1} .$$

Since $\rho > c$, the right-hand side is obviously positive. For $c = 0$, S_1 is defined to be equal to 0, so the inequality if satisfied.
For $c \geqq 1$, the right-hand side can be expanded as a geometric progression:

$$S_1 \leqq \frac{\rho^c}{c!} \frac{c}{\rho} \sum_{m=0}^{\infty} \left(\frac{c}{\rho}\right)^m = \frac{\rho^{c-1}}{(c-1)!} \sum_{m=0}^{\infty} \left(\frac{c}{\rho}\right)^m .$$

Using the definition of S_1 and splitting the summation on the right, we get

$$\sum_{n=0}^{c-1} \frac{\rho^n}{n!} \leqq \sum_{m=0}^{c-1} \frac{c^m}{(c-1)!} \rho^{c-1-m} + \frac{\rho^{c-1}}{(c-1)!} \sum_{m=c}^{\infty} \left(\frac{c}{\rho}\right)^m .$$

By changing the index of the first sum on the right ($n = c-1-m$) and bringing it over to the left, the relation can be written

$$\sum_{n=0}^{c-1} \left\{ \frac{1}{n!} - \frac{c^{c-1-n}}{(c-1)!} \right\} \rho^n \leqq \frac{\rho^{c-1}}{(c-1)!} \sum_{n=c}^{\infty} \left(\frac{c}{\rho}\right)^n . \qquad \text{(III-14)}$$

We will now show that the left-hand side of this inequality is negative. We must show

$$\frac{1}{n!} - \frac{c^{c-1-n}}{(c-1)!} = \frac{1}{n!}\left(1 - \frac{n!}{(c-1)!}\, c^{c-1-n}\right)$$

is negative; i.e., we must show that

$$\frac{n!\, c^{c-1-n}}{(c-1)!} > 1 ,$$

which is equivalent to

$$\frac{\overbrace{c \cdot c \; \ldots \; c}^{c-1-n \text{ terms}}}{(c-1)(c-2) \; \ldots \; (n+1)} > 1 ,$$

which is obviously true. Thus the left-hand side of (III-14) is negative. The sum on the right-hand side is obviously positive, so that the inequality holds. //

Therefore, since $(c!/\rho^c)S_1 + 1/[1-(\rho/c)] \leqq 0$, if the availability constraint is to be satisfied for a real y , $AS_2 + 1/[1-(\rho/c)]$ also must be less than or equal to zero. For $AS_2 + 1/[1-(\rho/c)] \leqq 0$, (III-11) can be written

$$\left(\frac{\rho}{c}\right)^{y-c} \geqq \left(\frac{c!}{\rho^c} S_1 + \frac{1}{1-\frac{\rho}{c}}\right) \Big/ \left(AS_2 + \frac{1}{1-\frac{\rho}{c}}\right) .$$

Since $\rho > c$, taking the natural log of both sides and dividing by $\ln(\rho/c)$, we get

$$(y-c) \geqq \ln\left[\left(\frac{c!}{\rho^c} S_1 + \frac{1}{1-\frac{\rho}{c}}\right) \Big/ \left(AS_2 + \frac{1}{1-\frac{\rho}{c}}\right)\right] \Big/ \ln\left(\frac{\rho}{c}\right) ,$$

or

$$y \geq c + \ln\left[\left(\frac{c!}{\rho^c} S_1 + \frac{1}{1-\frac{\rho}{c}}\right) \Big/ \left(AS_2 + \frac{1}{1-\frac{\rho}{c}}\right)\right] \Big/ \ln\left(\frac{\rho}{c}\right) .$$

This is an equivalent form of (III-12); therefore, again the boundary of the region satisfying the availability constraint for a given value of c is at the minimum integer value of y that satisfies the availability constraint. Let

$$y^* = c + \ln\left[\left(\frac{c!}{\rho^c} S_1 + \frac{1}{1-\frac{\rho}{c}}\right) \Big/ \left(AS_2 + \frac{1}{1-\frac{\rho}{c}}\right)\right] \Big/ \ln\left(\frac{\rho}{c}\right) .$$

Then for a given c, a boundary point of the region satisfying the availability constraint is given by the pair $(c, \lceil y^* \rceil)$. But y^* must be greater than or equal to c for (III-11) and the subsequent analysis, to be true. Therefore, since $\ln(\rho/c) > 0$,

$$\ln\left[\left(\frac{c!}{\rho^c} S_1 + \frac{1}{1-\frac{\rho}{c}}\right) \Big/ \left(AS_2 + \frac{1}{1-\frac{\rho}{c}}\right)\right]$$

must be greater than or equal to zero. This implies that

$$\frac{c!}{\rho^c} S_1 + \frac{1}{1-\frac{\rho}{c}} \geq AS_2 + \frac{1}{1-\frac{\rho}{c}} ,$$

or

$$\frac{c!}{\rho^c} S_1 \geq AS_2 ;$$

thus,

$$S_1 \geq \frac{\rho^c}{c!} AS_2 .$$

But this is the availability constraint for $y = c$. Therefore, for $c < \rho$, either the availability constraint is satisfied for $y \leq c$ or it is not satisfied at all. This property is useful in developing the algorithm for generating the boundary of the region satisfying the availability constraint, which is presented below.

Finally, for $y \geq c$, there is one more case of interest, and that is when $\rho = c$. For this case expression (III-11) is not valid.

Instead, using (III-3), we obtain

$$\sum_{n=0}^{c-1} \frac{\rho^n}{n!} + \frac{c^c}{c!} \sum_{n=c}^{y-1} \left(\frac{\rho}{c}\right)^n \geqq \frac{a}{1-a} \frac{\rho^c}{c!} \frac{\rho^{y-c}}{c^{y-c}} \sum_{j=0}^{M-1} \frac{(M-1)!}{(M-j)!} \left(\frac{\rho}{Mc}\right)^j .$$

Since $\rho = c$, $(\rho^{y-c}/c^{y-c}) = 1$ and the second summation on the left equals $(\rho^c/c!)(y-c)$. Substituting A , S_1 , and S_2 for the appropriate terms, we get

$$\frac{c!}{\rho^c} S_1 + y-c \geqq AS_2$$

or

$$y \geqq c + AS_2 - \frac{c!}{\rho^c} S_1 .$$

Let

$$y^* = c + AS_2 - \frac{c!}{\rho^c} S_1 .$$

Then for a given c , a boundary point of the region satisfying the availability constraint is given by the pair $(c, \lceil y^* \rceil)$. As for the case $c > \rho$, y^* will be greater than or equal to c if

$$AS_2 \geqq \frac{c!}{\rho^c} S_1 ,$$

which is equivalent to the situation in which the availability constraint $S_2 \geqq (\rho^c/c!)AS_1$ is not satisfied for $y = c$. Therefore, for $\rho = c$, either the availability constraint is satisfied for $y = c$ or there is a $y' > c$ such that (c, y') is the boundary point for the region satisfying the availability constraint for a specified c .

For $y \leqq c$, a convenient form of the availability constraint, similar to (III-11) for $y \geqq c$, has not been found.

From all of the above, it is now possible to state an algorithm that will identify all of the (c,y) pairs that define the boundary of the region satisfying the availability constraint for a given time period.

Algorithm.

```
1.  For the time period of interest, calculate
    $\rho = M(\hat{\lambda}/\mu)$ .
2.  Start with  $c = \lceil \rho \rceil$†.
3.  Calculate $AS_2$ , where $A = a/(1-a)$ and $S_2 = \sum_{j=0}^{M-1} \left(\frac{\rho}{Mc}\right)^j \frac{(M-1)!}{(M-1-j)!}$ .
4.  Calculate $S_1$ , where $S_1 = \sum_{n=0}^{c-1} \frac{\rho^n}{n!}$ .
5.  If $(c!/\rho^c)S_1 < AS_2$ , go to Step 12.
6.  If $c > \lceil \rho \rceil$ , go to Step 18.
7.  Decrement $c$ by 1 .
8.  Calculate $A$ , $S_1$ , $S_2$ .
9.  If $(c!/\rho^c)S_1 > AS_2$ , go to Step 7.
10. Increment $c$ by 1 .
11. Go to Step 18.
12. If $c \neq \rho$ , go to Step 15.
13. $y = \lceil c + AS_2 - (c!/\rho^c)S_1 \rceil$ .
14. Go to Step 16.
```

†$\lceil a \rceil$ is the smallest integer greater than a .

15. $$y = \left[c + \left\{\ln\left(\frac{c!}{\rho^c} S_1 + \frac{1}{1-\frac{\rho}{c}}\right) - \ln\left(AS_2 + \frac{1}{1-\frac{\rho}{c}}\right)\right\} \Big/ \ln\left(\frac{\rho}{c}\right)\right] .$$

16. (c,y) is a boundary point. Increment c by 1 .

17. Go to Step 3.

18. $y = c-1$.

19. Test if:

$$\sum_{n=0}^{y-1} \frac{\rho^n}{n!} \geqq A\rho^y \left\{\sum_{j=0}^{c-y-1} \frac{(M-1)!}{(M-1-j)!\,(y+j)!} \left(\frac{\rho}{M}\right)^j + \frac{c^c}{c!} \sum_{j=c-y}^{M-1} \frac{(M-1)!}{(M-1-j)!} \left(\frac{\rho}{M}\right)^j c^{-(y+j)}\right\} .$$

20. If the inequality is satisfied, go to Step 23.

21. The point $(c,y+1)$ is on the boundary. Increment c by 1 and continue the algorithm.

22. If $c = y+M+1$, the algorithm is terminated and all boundary points have been generated.

23. If $y = 1$, terminate the algorithm. All boundary points have been generated.

24. Decrement y by 1 .

25. Go to Step 19.

The purpose of each step of the algorithm will now be explained.

1. Calculate ρ , a parameter of the problem.

2. Initialize c equal to the greatest integer less than ρ . If there is a boundary point with $y > c$, then $c \geqq \rho$.

3. Calculate AS_2 , a term needed to determine if the availability constraint is satisfied for $y > c$.

4. Calculate S_1 , another term needed to evaluate the availability constraint for $y > c$.

5. Test if the boundary point is such that $y > c$.

6. The next boundary point has $y \leqq c$. Test if this is the first boundary point generated.

7. This is the first boundary point generated. Search for the minimum c such that the availability constraint is satisfied.

8. Calculate needed terms.

9. If the availability constraint is satisfied for $y < c$, then it will be satisfied for $y = c$. Test to see if the availability constraint is satisfied for $y = c$. If it is still satisfied, go to Step 7 to continue searching for the first c for which the availability constraint is not satisfied.

10. The first c for which the availability constraint is not satisfied has been determined. Set c to the last c for which the availability constraint can be satisfied.

11. Go to Step 18 to determine the boundary when $y < c$.

12. Test if c equals ρ .

13. Calculate the boundary y when $c = \rho$.

14. The boundary point is identified.

15. Calculate y for the boundary point when $c > \rho$ and $y \geqq c$.

16.⎫ The boundary point is determined for $y \geq c$.
17.⎭ Proceed to calculate the next boundary point.

18. Boundary point calculation for $y \leq c$.

19. Determine whether the availability constraint is satisfied.

20. Test if availability constraint is satisfied. If it is satisfied, continue to decrement y for the specified c to determine the minimum y .

21. The availability constraint is no longer satisfied for the particular c . Therefore, the previous $(y,y+1)$ was the minimum y for the specified c for which the availability constraint was satisfied. Proceed to the next c .

22. If c is greater than $y+M$ we know that there are no more boundary points. This is because of the physical reality as well as the mathematical formulation. There will never be more repair channels than there are machines in operation plus spares.

23. We will never have negative spares, so if $y = 0$ we will not have any additional boundary points.

24.⎫ Continue the search for the minimum y that
25.⎭ satisfies the availability constraint for a specified c .

The algorithm is best illustrated by Figures III-2 and III-3. We start by choosing $c = \lceil \rho \rceil$. If the availability constraint is satisfied for some y less than or equal to c we are on Figure III-3; otherwise we are using Figure III-2. For Figure III-2 we proceed from $c = \lceil \rho \rceil$ and increment c by 1 , finding the minimum y that satisfies the availability constraint for each c . Eventually the boundary crosses the line $y = c$. To the right of this

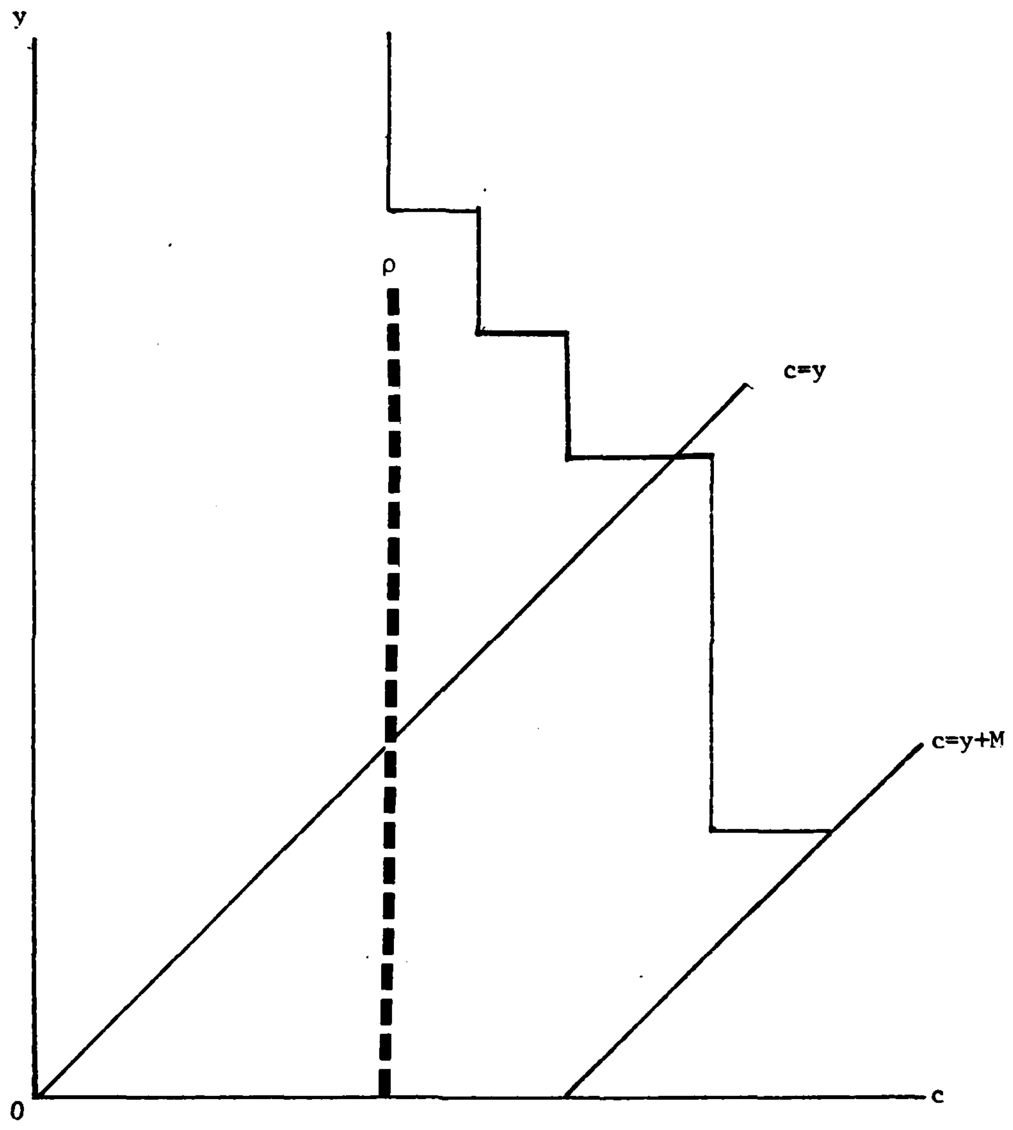


Fig. III-2. Boundary of region satisfying availability constraint when there are boundary points (c,y) where $y > c$.

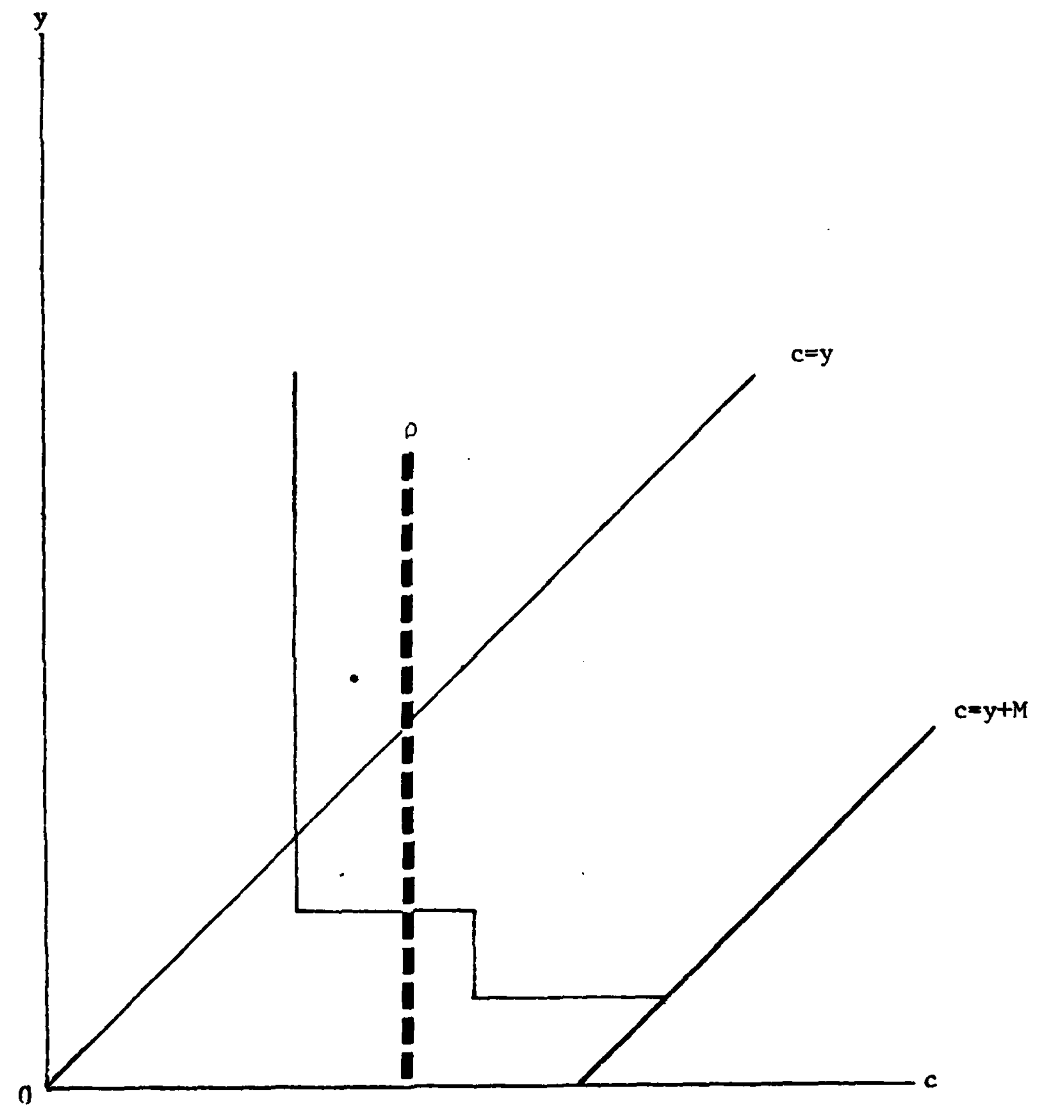


Fig. III-3. Boundary of region satisfying availability constraint when all boundary points (c,y) satisfy $y < c$.

line, using Steps 18-25, y is decremented by 1 and the minimum c that satisfies the availability constraints for the specified y is determined by incrementing. This procedure continues until either $y = 1$ or the availability constraint is not satisfied for $c = y+M$. For Figure III-3 there is no boundary point for $y > c$. We proceed from $c = \lceil\rho\rceil$ to the minimum c for which there is a y such that the availability constraint is satisfied. Once this c is determined we proceed in the same manner as above for the cases when $y < c$.

This procedure, therefore, is one in which the minimum c for which the availability constraint is satisfied is determined, and by incrementing c until either $c = y+M$ or $y = 1$, the minimum y associated with each c for which the availability constraint is satisfied, is determined. The algorithm starts from the upper left of the region and proceeds until the boundary crosses either the abscissa or the line $c = y+M$. The initial c either equals the greatest integer less than ρ or lies below the line $y = c$. Therefore, if ρ is finite, the initial c is finite. If AS_2 is finite, then the initial y is finite. Therefore, the boundary of the region satisfying the availability constraint can be described by a finite number of (c,y) pairs.

Theorem 4: If ρ and AS_2 are finite, the boundary of the region satisfying the availability constraint can be enumerated by a finite number of (c,y) pairs.

Proof: For a finite ρ , the minimum c , say c^* , for which the availability constraint is satisfied is finite, $c^* \leqq \rho$. If $c^* = \rho$ and AS_2 is finite, the value of y , say y^* , corresponding to the boundary is given by (III-12) and is finite. For $c^* < \rho$ we have shown that y must be less than c^* . Therefore, the y^* associated with c^* , the minimum c for which the availability constraint is satisfied, is finite.

We have shown that when a pair (c,y) satisfies the availability constraint, then the pair (c',y'), where $c' \geqq c$ and $y' \geqq y$, also satisfies the availability constraint. Therefore, if a pair (c^+,y^+) is on the boundary, then either $c^+ < c$ or $y^+ < y$. But c^* is the minimum c for which the availability constraint is satisfied. Therefore, for any other boundary point satisfying the availability constraint, it must be true that $c^+ \geqq c^*$. So, for all other boundary points $y^+ < y^*$.

Then we have $y^* > y > 0$ and y is integer. Since y^* is finite for AS_2 finite, there are a finite number of points needed to enumerate the boundary of the region satisfying the availability constraint. //

The use of this algorithm provides a method of enumerating the boundary of the region satisfying the availability constraint for each time period. We know that for a given c (or a given y) the minimum y (c) for which the availability constraint is satisfied is on the boundary. If the integer constraint and the availability constraints were the only constraints of the problem with a linear objective function, the solution would be on this boundary in each time period. There are, however, two additional types of constraints, $c_i \geqq c_{i-1}$ and $y_i \geqq y_{i-1}$. The enumeration scheme must account for these constraints as well. To handle these constraints a construct called a "corner" will be defined. This is described next.

III.5 Corners

In this section we will define a quantity which we term a "corner" and describe its properties. Through the use of corners, the enumeration of the overall feasible region is achieved.

Definition: For a given point $a = (a_1, a_2, \ldots, a_m)$ the *corner generated by* a is

$$A = \{x \geqq a\} ,$$

where $x = (x_1, x_2, \ldots, x_m)$.

Given points $a^1, \ldots, a^p$ which generate corners $A^1, \ldots, A^p$, we define a *dominated region* or *D region* as $D = \bigcup_i A^i$.

Thus D contains all points contained in at least one of the corners.

Given any region R of the form D above, there is a unique minimum set of points $G(R)$ which generates R ; i.e., $a^i \varepsilon G(R)$ if and only if $a^i \varepsilon R$, but a^i is not dominated by another point in R .

Theorem 5: Let A^i and A^j be corners generated by a^i and a^j . Then $A^i \cap A^j$ defines a corner A^k generated by a^k , where $a^k_n = \max(a^i_n , a^j_n)$, $n=1,2$.

Proof: By definition $A^k = \{x \geqq a^i$ and $x \geqq a^j\}$. Therefore, if $x \varepsilon A^k$, we must have $x_n \geqq \max\{a^i_n , a^j_n\}$. In particular, the point a^k defined by $a^k_n = \max(a^i_n , a^j_n)$ is in A^k . Clearly no other point in A^k dominates a^k , so that a^k generates A^k .

Figure III-4 illustrates this property of corners. There are two separate presentations given in Figure III-4: A and B. In both, there are two corners generated by two points: a^1 and a^2 . In Figure III-4A we see that a D region is generated using both points a^1 and a^2 , and the intersection $A^1 \cap A^2$ is generated

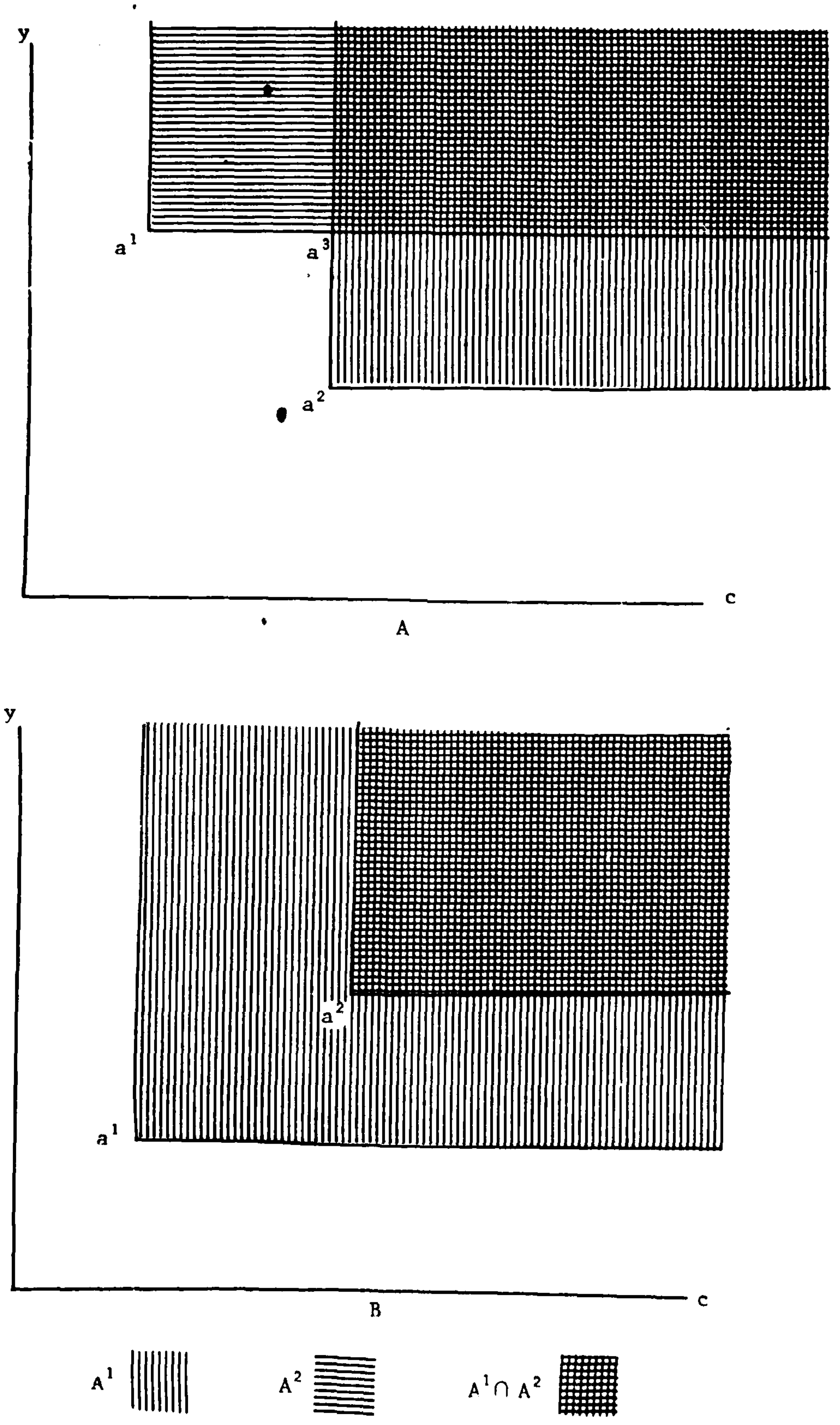


Fig. III-4. $A^1 \cap A^2$.

by a new corner generated by a^3 . In Figure III-4B, the D region is the same as the corner A^1 generated by a^1 , and $A^1 \cap A^2$ is A^2 , the same as the corner generated by a^2 .

Another property of D regions, which will prove useful, is the manner in which the intersection of two D regions can be generated by the generators of each of the D regions. Let $D_1 = \bigcup_{i\varepsilon I} A_1^i$ and $D_2 = \bigcup_{j\varepsilon J} A_2^j$. Then

$$\begin{aligned} D_1 \cap D_2 &= \left(\bigcup_{i\varepsilon I} A_1^i\right) \cap \left(\bigcup_{j\varepsilon J} A_2^j\right) \\ &= \bigcup_{i\varepsilon I\,,\,j\varepsilon J} \left(A_1^i \cap A_2^j\right) \end{aligned}$$

This property can be visualized from Figure III-5. The dominated region D_1 is generated by the set of corners (A^1,A^2,A^3) associated with the points (a^1,a^2,a^3) , and the dominated region D_2 is generated by the set of corners (B^1,B^2,B^3) associated with the points (b^1,b^2,b^3) . The new region, denoted by $\widetilde{D}$, is generated by the set of corners (C^1,C^2,C^3,C^4,C^5) generated by the points (c^1,c^2,c^3,c^4,c^5) , determined via Theorem 5. Note that some pairs (e.g., a^1 and b^2) produce a corner which is contained in some other corner (e.g., C^1).

The utility of corners arises from the properties of the region satisfying the availability constraint, as discussed in the previous section. As stated in Theorem 2, if (c',y') satisfies the availability constraint then (c,y) satisfies the constraint where $c \geq c'$ and $y \geq y'$. Therefore, if the point (c',y') satisfies the availability constraint, all of the pairs contained in the corner generated by the point (c',y') also satisfy the availability constraint.

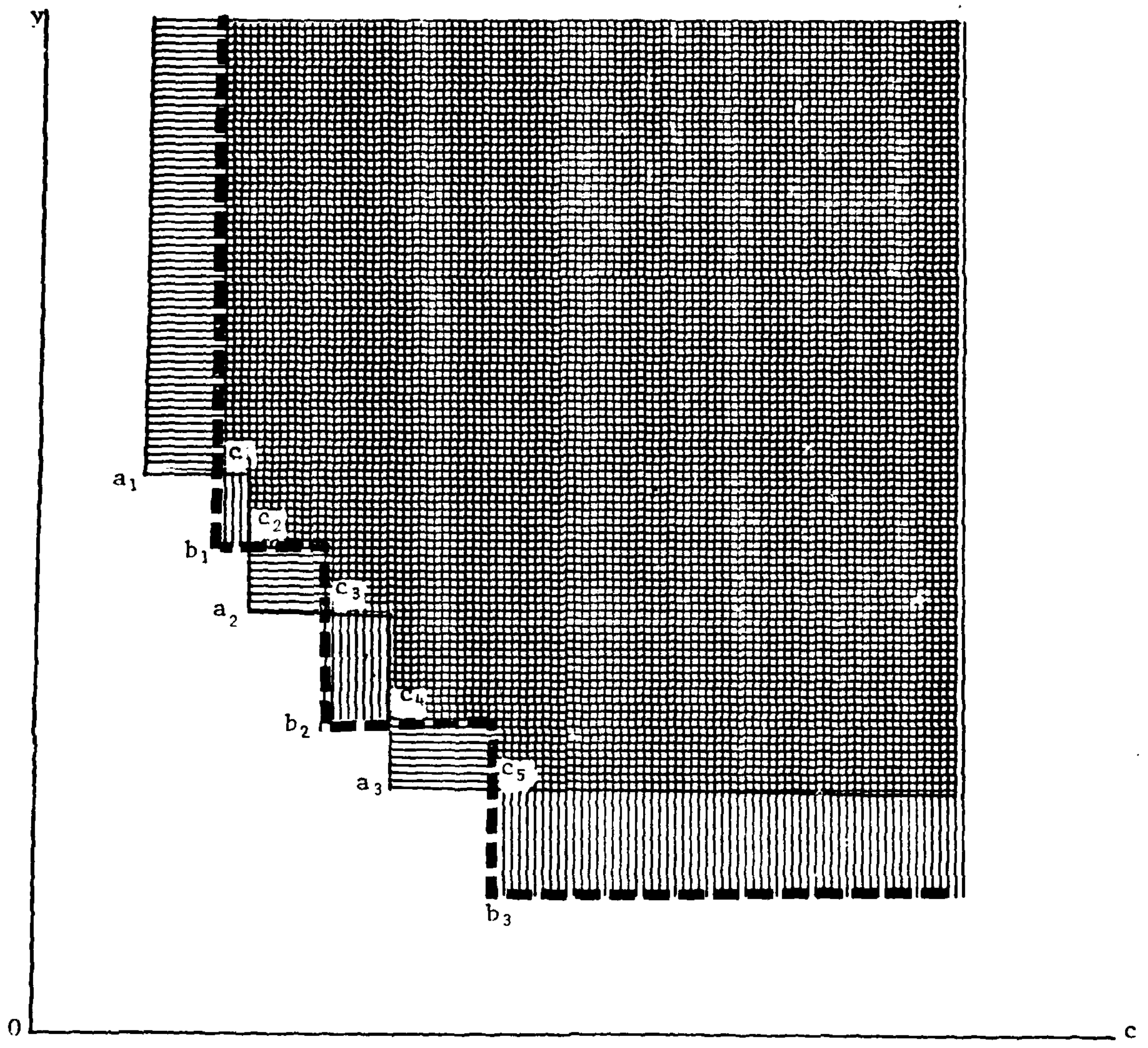


Fig. III-5. $D_1 \cap D_2$.

D_1	▤
D_2	▥
$\tilde{D}$	▦

For any given time period i, the region satisfying the availability constraint can be described by a D region, D_i. The boundary of D_i has the property that for (c_i', y_i') on the boundary of D_i then either (a) for y_i' fixed, c_j' is the smallest integer c_i for which (c_i', y_i') satisfies the availability constraint for time period i; or (b) for c_i' fixed, y_i' is the smallest integer y_i for which (c_i', y_i') satisfies the availability constraint. If both properties (a) and (b) are satisfied, then (c_i', y_i') generates a corner composing the D region D_i. The enumeration algorithm, specified in the previous section, generates the unique minimum set of points $G(D_i)$ for region D_i.

We have defined the D region, D_i, that contains all of the points that satisfy the availability constraint for the i*th* time period. We have also specified the properties of the boundary of D_i and presented a method of enumerating the points that generate the corners composing D_i. Besides the integer constraint and the availability constraint, there remain the monotonicity constraints, i.e., $c_{i+1} \geqq c_i \geqq c_{i-1}$ and $y_{i+1} \geqq y_i \geqq y_{i-1}$. We can use the properties of D regions to eliminate points that do not satisfy these constraints.

Theorem 6: If the pair (c_i, y_i) satisfies the availability constraint for time period i, then for (c_i, y_i) to be feasible it must also satisfy the availability constraint for time period j, for all j less than i.

Proof: By the monotonicity constraints, $c_i \geqq c_{i-1} \geqq \cdots \geqq c_1$ and $y_i \geqq y_{i-1} \geqq \cdots \geqq y_1$.

From Theorem 2 we know that if (c_j, y_j) satisfies the availability constraint in time period j, then (c_j', y_j'), $c_j' \geqq c_j$ and $y_j' \geqq y_j$, must also satisfy the availability constraint for time period j.

Therefore, since $c_i \geqq c_j$, $j \leqq i$ and $y_i \geqq y_j$, $j \leqq i$, then the pair (c_i, y_i) must satisfy the availability constraint for the time period i for all $j \leqq i$. //

Therefore, if the point associated with the pair (c_i, y_i) that satisfies the availability constraint for time period i is feasible, it must not only be contained in D_i, but must also be contained in D_j for all j less than i. This implies that the point (c_i, y_i) is contained in the intersection of D_i and D_{i-1}; i.e., $(c_i, y_i) \varepsilon D_i \cap D_{i-1}$.

Define $\tilde{D}_1 = D_1$ (there is no time period previous to the first), and set $\tilde{D}_i = D_i \cap \tilde{D}_{i-1}$. Then for a point (c_i, y_i) to be feasible, i.e., c_i and y_i integer, (c_i, y_i) satisfies the availability constraint, and c_i and y_i satisfy the monotonicity constraints, it must not only be contained in D_i but it must also be contained in $\tilde{D}_i$. Now $\tilde{D}_i$ can be explicitly determined using the property of the intersection of two D regions and through the use of Theorem 5.

Let us further examine the nature of the boundary of $\tilde{D}_i$.

Theorem 7: Let $(\tilde{c},\tilde{y})$ be on the boundary, $\partial\tilde{D}_i$, of $\tilde{D}_i$. Then

(a) for $\tilde{y}$ fixed, $\tilde{c}$ is the smallest integer c for which $(c,\tilde{y})$ satisfies the j*th* availability constraint for some $j \leq i$; or ,

(b) for $\tilde{c}$ fixed, $\tilde{y}$ is the smallest integer y for which $(\tilde{c},y)$ satisfies the k*th* availability constraint for $k \leq i$.

Proof: The proof is by induction.

For $j = 2$, let $(\tilde{c},\tilde{y}) \in \partial\tilde{D}_2$. Then either $(\tilde{c},\tilde{y}) \in \partial D_1$ or $(\tilde{c},\tilde{y}) \in \partial D_2$, since $\tilde{D}_2 = D_1 \cap D_2$. Therefore, $(\tilde{c},\tilde{y})$ satisfies either (a) or (b) for time period 1 or 2, respectively. The theorem is therefore true for $j = 2$ (it is obviously true for $j = 1$).

Then let us assume it is true for $i = j$. We must show that it is true for $k = j+1$. Let $(\tilde{c},\tilde{y}) \in \partial\tilde{D}_{j+1}$. Then either $(\tilde{c},\tilde{y}) \in \partial D_{j+1}$ or $(\tilde{c},\tilde{y}) \in \partial\tilde{D}_j$, since $\tilde{D}_{j+1} = D_{j+1} \cap \tilde{D}_j$. If $(\tilde{c},\tilde{y}) \in \partial D_{j+1}$, then (a) or (b) is true by the definition of D_{j+1} .

If $(\tilde{c},\tilde{y}) \in \partial\tilde{D}_j$, the theorem holds by the inductive assumption. //

Theorem 7 implies that a point $(\tilde{c}_i,\tilde{y}_i)$ on the boundary of $\tilde{D}_i$ is on the boundary of the D region satisfying the availability constraint D_i , or that a monotonicity constraint is binding; i.e., $\tilde{c}_i = \tilde{c}_{i-1}$ or $\tilde{y}_i = \tilde{y}_{i-1}$.

As was stated previously, a linear objective function will achieve its minimum at an extreme point of the convex set of all possible solutions. This set is the feasible region as determined by the constraints. The spares provisioning problem, as formulated, has three types of constraints: the integer constraint, the availability constraint, and the monotonicity constraints. The extreme points associated with the integer and availability constraints are contained in the set of points generating the corners of the D regions, D_i. The inclusion of the backward monotonicity constraint, i.e., $c_i \geqq c_{i-1}$ and $y_i \geqq y_{i-1}$, is accounted for in the generation of the $\tilde{D}_i$. There remains only one type of constraint on c_i and y_i that has not been accounted for in the generation of the $\tilde{D}_i$. These are the forward monotonicity constraints, i.e., $c_i \leqq c_{i+1}$ and $y_i \leqq y_{i+1}$. By this it is meant that if $c_i = c_{i+1}$ or $y_i = y_{i+1}$, there is no assurance that the associated point is on the boundary of $\tilde{D}_i$. This last constraint, however, can be accounted for in the dynamic programming procedure, as discussed in the next section.

For the last time period there is no such constraint. Therefore, the extreme points for the last time period, ℓ, must come from the set of points that generate the corners of $\tilde{D}_\ell$. This is a finite set. For all other time periods, $j < \ell$, there is a finite set of points that generate the $\tilde{D}_j$ regions. These points contain the set of extreme points that satisfies the availability constraint, integer constraint, and backward monotonicity constraint, i.e., $c_j \geqq c_{j-1}$ and $y_j \geqq y_{j-1}$.

III.6 Dynamic Programming

As dynamic programming is a general problem solving approach, there does not exist a standard mathematical formulation. However, there are certain characteristics associated with a problem if dynamic programming can be applied. They are:

1. The problem can be divided into stages with a policy decision required at each stage.
2. There are a number of states associated with each stage.
3. The effect of the policy decision at each stage is to transform the current state into a state associated with the next stage.
4. Given the current state, an optimal policy for the remaining stages is independent of the policy adopted in the previous stages.
5. The solution procedure begins by finding the optimal policy for each state of the last stage.
6. One can establish a recursive relationship that relates the optimal policy for each state at the current stage to the given optimal policy for each state at the next stage.
7. The solution procedure moves backwards, stage by stage, until the optimal policy starting with the first stage is found [9].

At first glance the spares provisioning problem appears to satisfy all of these characteristics. There are identifiable stages associated with each time period. The states are associated with the choice of the combination of the number of spares and repair channels at each time period. The linear objective function provides a relationship upon which to determine the optimal policy at each stage.

On second thought, the problem becomes formidable. There are two major obstacles to the direct application of dynamic programming techniques to the spares provisioning problem. First, the states of the problem cannot be specified for any but the last and next to last stages (time periods 1 and 2), because the $\overline{\lambda}_i$'s cannot be determined until subsequent stages are solved. Secondly, for each stage there is potentially an infinite number of states [(c,y) pairs that satisfy the constraints].

As has been discussed, $\bar{\lambda}_i$ for $i > 2$ is dependent on the choice of c_j and y_j, where $j < i$. In a standard dynamic programming solution the problem is solved working backwards in time, i.e., the first stage solved is associated with the last time period. Therefore, $\bar{\lambda}_i$, for a particular time period, cannot be determined until all (c_j, y_j) pairs are specified for $j < i$. Although most dynamic programming solutions do proceed backwards in time, there are techniques for proceeding forward [10]. These, however, require knowledge of the initial state of the problem. For the spares provisioning problem, however, there is potentially an infinite number of initial states.

This first problem, the indeterminateness of $\bar{\lambda}_i$, can be resolved by using the branch and bound procedure presented previously. Through the branch and bound procedure, subproblems are created in which the $\bar{\lambda}_i$'s are replaced by $\hat{\lambda}_i$. The $\hat{\lambda}_i$ are independent of the choice of (c_j, y_j) for all time periods. Therefore, it is possible to determine all states at each stage of the problem, even when proceeding backwards in time.

The second problem, the number of potential states, can be eliminated by the enumeration technique and the use of corners. There is an infinite number of choices of c and y that satisfy the four types of constraints: c and y are integer; the availability constraint; $c_i \leqq c_{i+1}$, $y_i \leqq y_{i+1}$; and $c_i \geqq c_{i-1}$, $y_i \geqq y_{i-1}$. We know, however, due to the linearity of the objective function, that the solution will lie on the extreme points of the boundary of the region satisfying these constraints. The enumeration technique accounts for the integer constraint and determines the boundary of the region satisfying the availability constraint. The technique associated with the corners modifies the boundary due to the backward monotonicity constraints $(c_i \geq c_{i-1}, y_i \geq y_{i-1})$. The only constraints not accounted for by techniques already specified are the forward monotonicity constraints $(c_i \leq c_{i+1}, y_i \leq y_{i+1})$ which are accounted for directly in the dynamic programming procedure. In determining the set of extreme points on the boundary satisfying the first three of the four constraint types for each time period, we end up with a finite, and usually small, set of points.

The forward monotonicity constraints, $c_i \leq c_{i+1}, y_i \leq y_{i+1}$, do not affect the use of dynamic programming. These constraints relate the optimal choice of c_i and y_i in a particular time period to the selection of c_{i+1} and y_{i+1} in the next time period. Since the dynamic programming procedure works backward in time, an optimal choice of c_{i+1} and y_{i+1} is made before the optimal c_i and y_i are determined. Thus once the optimal policy is found for a particular stage (associated with time period $i+1$), the states of the next stage (associated with time period i) can be specified. For the first stage (associated with the last time period), since there is no next time period, the states are not constrained by the fourth type of constraint. Therefore, the states of the first stage of the problem are given as a result of finding the $\tilde{D}_\ell$ region where ℓ is the last time period.

For this problem there is a set of states associated with the last time period. Therefore, it is necessary to determine a set of optimal policies, each associated with a state of the first stage (last time period). From this set of optimal policies, the best policy (lowest cost) is the true optimum. In practice this is accomplished by solving k dynamic programming problems, where k is the number of (c_ℓ, y_ℓ) pairs that form the set of extreme points of the region that satisfy the integer, availability, and backward monotonicity constraints for the last time period.

At this point, all of the techniques needed to determine the exact solution to the spares provisioning problem have been discussed. These can be summarized as follows.

1. Use a branch and bound procedure based upon the approximation of $\overline{\lambda}_i$ by $\hat{\lambda}_i$ $(\hat{\lambda}_i \leqq \overline{\lambda}_i)$. The $\hat{\lambda}_i$'s are independent of the (c_j, y_j) , $j < i$, whereas the $\overline{\lambda}_i$'s are not.

2. Enumerate the convex hull of the region satisfying the availability constraint and integer constraint for each time period.

3. Use corners to determine the $\tilde{D}$ regions. This provides a method of enumerating the (c_i, y_i) pairs associated with the convex hull of the region satisfying the availability constraint for time period i and the constraints $c_i \geqq c_{i-1}$, $y_i \geqq y_{i-1}$.

4. By dynamic programming, obtain the solution to each of the subproblems associated with the branch and bound procedure.

The next chapter will discuss the implementation of these techniques and provide computational experience associated with implementation.

CHAPTER IV

COMPUTATIONAL EXPERIENCE

This chapter discusses the applicability and utility of the approach to the optimization of the spares provisioning problem described in the preceding chapters. All of the techniques discussed in Chapter III have been incorporated into a computer program. A listing of this program is contained in Appendix A. After a discussion of the structure of the program and the function of its elements, example problems are presented. The nature of the problems and their solution are then discussed.

The computer program was written to be run on the Hewlett Packard 3000 computer of the School of Engineering and Applied Science of The George Washington University. This is a time-sharing system that allows direct input of data at the time the program is running. This latter capability is utilized in performing the branch and bound procedure.

The program consists of the main routine and three subroutines. The main routine handles the data input and the output of results, as well as the logical flow of the solution. Subroutine QUE calculates all terms associated with the queueing aspects of the problem. This includes the availability constraint and the mean number repaired. Subroutine UNION calculates the intersection of dominated regions between two time periods. The last subroutine, DP, generates the optimal solution of a branch and bound subproblem through dynamic programming techniques.

Inputs to the program for each time period are the failure rate, λ_i ; the mean time of repair, $1/\mu_i$; the costs $C_{1,i}$, $C_{2,i}$, $C_{3,i}$, and $C_{4,i}$; and the availability requirement and discount factor. As discussed in Chapter II, only the costs $C_{1,i}$, the cost associated with each repair channel, and $C_{2,i}$, the cost associated with each spare, are used in the evaluation of the objective function. However, for each solution, the actual cost based upon $C_{1,i}$, $C_{2,i}$, $C_{3,i}$, and $C_{4,i}$ is also calculated and printed out.

From the input, the $\hat{\lambda}_i$'s are calculated. These are lower bounds on the $\overline{\lambda}_i$'s , i.e., $\hat{\lambda}_i \leqq \overline{\lambda}_i$, associated with the extended problem of the branch and bound procedure. A solution is obtained using the $\hat{\lambda}_i$'s and then, based upon the optimal policy, the corresponding $\overline{\lambda}_i$'s are calculated. If the solution satisfies all of the constraints when they are evaluated using the exact $\overline{\lambda}_i$'s , then the solution is feasible and the problem is solved. If all of the availability constraints are not satisfied using the exact $\overline{\lambda}_i$'s , then the solution is infeasible and branching must therefore occur.

The initial experience in the development of this program indicated that, for the data associated with the gas turbine spares provisioning problem, the solution to the first subproblem of the branch and bound procedure was usually feasible. Therefore, instead of devising a complex computer program that would generate the complete branch and bound tree, it was decided that the branching procedure could best be handled external to the computer, using the computer's interactive capability to input the parameters associated with each new branch. A later modification to the program allows the computation of the solution of a subproblem and the set of branches that directly emanate from that subproblem.

Because the complete branch and bound structure is not computed within the program, the Best Lower Bound (BLB) is not automatically determined. The program prints out the optimal value of the objective function and the value of the total cost, as well as the optimal policy for each of the branch and bound problems. It is noted if the solution is feasible. The (c_k, y_k) pairs and the associated $\overline{R}_k$ are output. The program does keep track of the Best Upper Bound (BUB).

Based upon the output, it is up to the operator to determine the BLB. The operator must then input the next problem to be solved. This is accomplished by specifying the branching time period and the policy for the previous time periods. The policy for the first time period does not have to be specified because the program automatically determines it.

Based upon this new input problem, the program evaluates a solution, determines if it is feasible, branches on the (c_k, y_k) pairs, and solves the new problems formed by the branching. If the solution to a problem is feasible, no further branching will occur from this problem. If the solution to a feasible problem is less than the BUB, the BUB is set equal to the value of the objective function for this new feasible solution. If the value of the objective function for a problem, whether it is feasible or not, is greater than the BUB, no further branching need occur from this problem. This is because the theory of branch and bound states that neither this problem nor any of its branches can yield the solution.

The procedure is best illustrated through an example. Three test problems have been created. Each problem has five time periods with identical M_i's , λ_i's , and $1/\mu_i$'s ; they differ only in [illegible] costs. Table IV-1 displays the values used for each M_i , [illegible] $_i$.* The real costs and the discounted costs are given

[illegible] the computer program, the λ_i's were chosen to [illegible] time as opposed to most actual situations, where [illegible] improves with time.

TABLE IV-1

INPUT PARAMETERS FOR EXAMPLE PROBLEMS A, B, C

M_i	λ_i	$1/\mu_i$
10	.00050000	50.0000
20	.00060000	50.0000
30	.00070000	50.0000
40	.00070000	50.0000
50	.00070000	50.0000

for each of the three problems in Table IV-2. It should be noted that for Problem A, $C_{1,i} = C_{2,i} = C_{3,i} = C_{4,i}$; for Problem B, $.5C_{2,i} = C_{1,i} = C_{3,i} = C_{4,i}$; and for Problem C, $.5C_{1,i} = C_{2,i} = C_{3,i} = C_{4,i}$. The computer model was used to obtain the exact solution that minimized the objective function

$$\sum_{i=1}^{n} d^i \left[C_{1,i}(c_i - c_{i-1}) + C_{2,i}(y_i - y_{i-1})\right] .$$

This is labelled OBJ in the subsequent figures. The total cost, including the terms $C_{3,i}$ and $C_{4,i}$, is labelled COST in these figures.

Figure IV-1 shows the branch and bound structure associated with Problem A. At the top of this figure is the extended problem. This problem is solved using $\hat{\lambda}_i$, where $\hat{\lambda}_i \leq \overline{\lambda}_i$. The solution is tested for feasibility using the $\overline{\lambda}_i$ calculated from the specified policy. It is seen that this solution is not feasible. Since $\hat{\lambda}_1 = \overline{\lambda}_1 = \lambda_1$, it is not necessary to branch at the first time period. For this problem there are two pairs, (1,4) and (2,3). These pairs

TABLE IV-2

COSTS FOR EXAMPLE PROBLEMS A, B, C

Problem A

Costs			
C_1	C_2	C_3	C_4
10.00	10.00	10.00	10.00
10.00	10.00	10.00	10.00
10.00	10.00	10.00	10.00
10.00	10.00	10.00	10.00
10.00	10.00	10.00	10.00
Discount Factor=.1		Availability=.9	
10.00	10.00	10.00	10.00
9.09	9.09	9.09	9.09
8.26	8.26	8.26	8.26
7.51	7.51	7.51	7.51
6.83	6.83	6.83	6.83

Problem B

Costs			
C_1	C_2	C_3	C_4
10.00	20.00	10.00	10.00
10.00	20.00	10.00	10.00
10.00	20.00	10.00	10.00
10.00	20.00	10.00	10.00
10.00	20.00	10.00	10.00
Discount Factor=.1		Availability=.9	
10.00	20.00	10.00	10.00
9.09	18.18	9.09	9.09
8.26	16.53	8.26	8.26
7.51	15.03	7.51	7.51
6.83	13.66	6.83	6.83

Problem C

Costs			
C_1	C_2	C_3	C_4
20.00	10.00	10.00	10.00
20.00	10.00	10.00	10.00
20.00	10.00	10.00	10.00
20.00	10.00	10.00	10.00
20.00	10.00	10.00	10.00

TABLE IV-2--*continued*

Discount	Factor=.1	Availability=.9	
20.00	10.00	10.00	10.00
18.18	9.09	9.09	9.09
16.53	8.26	8.26	8.26
15.03	7.51	7.51	7.51
13.66	6.83	6.83	6.83

are associated with the corners defining the dominated region of the second time period.

Two new problems are created. In Problem 2, the solution must contain the policy $c_2 = 1$, $y_2 = 4$. The exact $\overline{\lambda}_3$ is calculated based upon this policy. The subsequent $\overline{\lambda}_i$'s ($\overline{\lambda}_4$ and $\overline{\lambda}_5$) are approximated by $\hat{\lambda}_i$ such that $\hat{\lambda}_4 \leqq \overline{\lambda}_4$ and $\hat{\lambda}_5 \leqq \overline{\lambda}_5$. For Problem 3 the same procedure is used except $c_2 = 2$ and $y_2 = 3$. The exact $\overline{\lambda}_3$ is calculated on this policy and $\hat{\lambda}_i$'s are used for time periods 4 and 5. Problems 2 and 3 are then solved and tested for feasibility.

We see that neither Problem 2 nor 3 is feasible. Our BLB is 70.79 and there is no BUB. Problems 1, 2, and 3 are automatically solved in one pass of the computer run.

Since the BLB's are the same for Problems 2 and 3, but the lower cost is associated with Problem 2, we branch on this problem. The computer program allows us to input the specified policy for time periods 2 and 3. It is not necessary again to input the values for the costs, M_i's , λ_i's , or $1/\mu_i$'s .

From Problem 2, in time period 2 there are three possible policies, (1,17), (2,4), and (3,4). The computer next solves Problem 4, which was created by inputing $c_2 = 1$, $y_2 = 4$, $c_3 = 1$, and

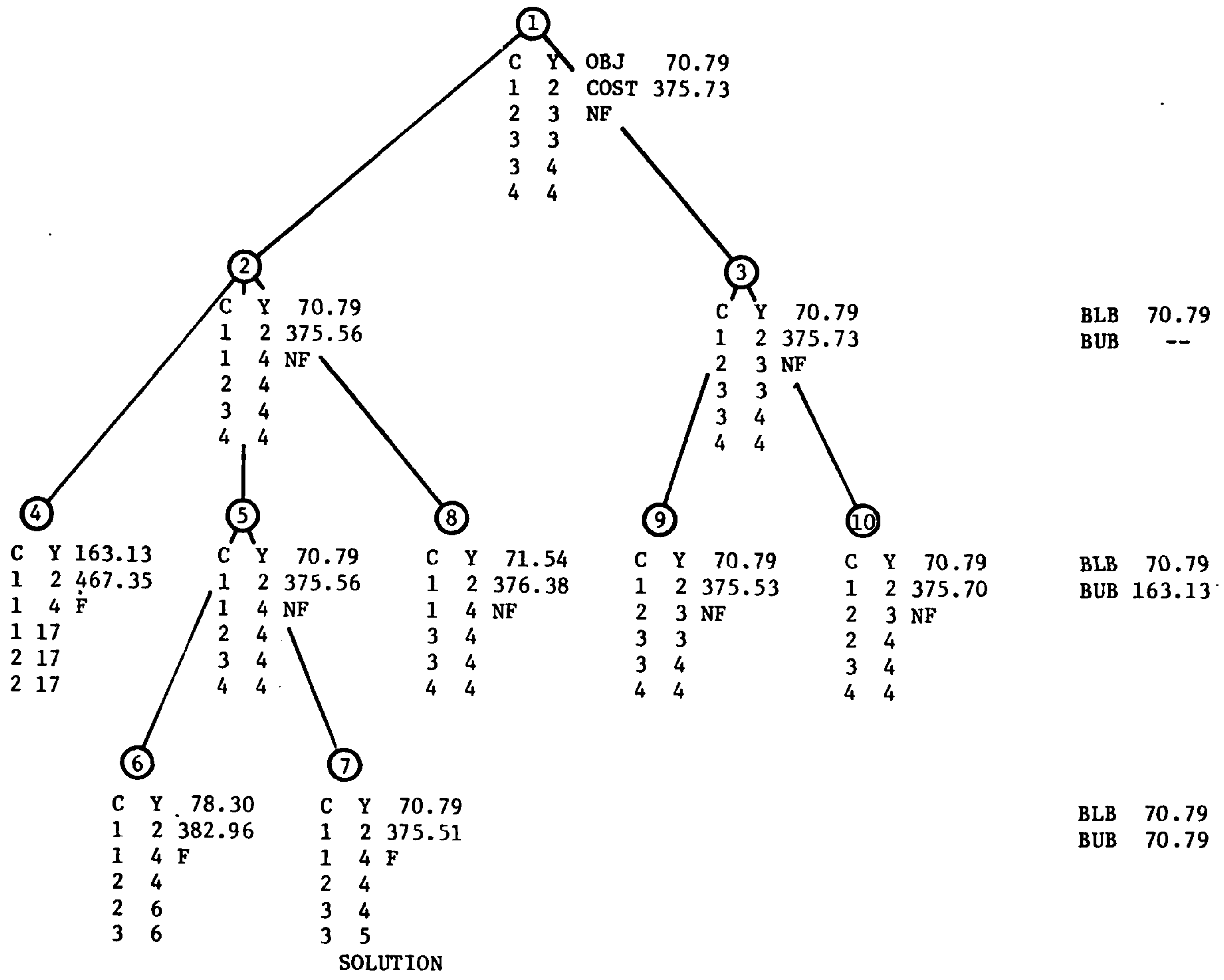


Fig. IV-1. Extended branch and bound tree for Problem A.

$y_3 = 17$. Based upon this policy $\overline{\lambda}_4$ is calculated exactly and $\hat{\lambda}_5$ is determined. The computer solves this problem. Since it is feasible, no further branching is necessary.

Problem 5 is then input to the computer. For this problem, $c_2 = 1$, $y_2 = 4$, $c_3 = 2$, and $y_3 = 4$ are specified. The exact $\overline{\lambda}_4$ is calculated and $\hat{\lambda}_5$ determined for this policy. A nonfeasible solution is found and two more branches are automatically generated. We see that the solutions to Problems 6 and 7 are both feasible. For Problem 7 the value of the objective function is equal to the BLB. Since this is a feasible solution we know that it is an exact solution to the original problem.

At this point the branch and bound procedure could be terminated. There may be other solutions that are as good as Problem 7, but there will be none better. For the sake of completeness, the procedure is continued. Problems 8, 9, and 10 are input sequentially to the computer. Since none of them gives feasible solutions that are better than that of Problem 7 (which determined the BUB), no further branching occurs.

We therefore have an exact optimal solution to Problem A given by Problem 7. The minimum value of the objective function is 70.79 and the cost associated with this solution is 375.51.

Since this procedure attempts to minimize the objective function which is associated with only the direct cost of the repair channels and spares (terms associated with the $c_{1,i}$ and $c_{2,i}$), it is not necessary that the total cost (including $c_{3,i}\overline{R}_i$) associated with the solution be the minimum possible total cost. For this problem the solution provides not only the minimum of the objective function, but the minimum total cost for all of the subproblems calculated.

Problem B is similar to Problem A except that the cost of the spares has been doubled. A branching procedure similar to that of the previous problem is again followed. As seen in Figure IV-2, at the end of the first two stages we have no feasible solutions and the BLB is 107.29. Since the BLB is associated with Problem 3 we branch on it. When Problem 4 is input, the program automatically generates the solutions to Problems 4, 5, and 6. We next input Problem 7, generating solutions to Problems 7 and 8. This exhausts the branches from Problem 3.

We next branch on Problem 2 by inputting Problem 9. This has a feasible solution so no branching comes from it. We then input Problem 10 which branches into Problems 11 and 12. Finally, we input Problem 13 with its branch, Problem 14.

After all 14 problems have been solved, each of the solutions to the unbranched problems is feasible. Therefore, the BLB equals the BUB and the solution is associated with the problem that gives the BU(L)B. Problem 8 gives the best solution and again, serendipitously, it also has the lowest total cost.

From Figure IV-3 we see that Problem C requires the solution of only one problem. For this case the solution to the extended problem is feasible and therefore an exact optimal to the given problem.

It is of interest to compare the solutions to the three problems. Since all three problems use the same values for the M_i's , λ_i's , and $1/\mu_i$'s , the feasible regions are identical. Therefore the enumeration of potential solution policies is the same for each problem. The only reason the solutions will differ is because of the differences in the objective function.

Comparing the solutions for Problem A and Problem B (Figures IV-1 and IV-2, respectively), we see that the policies for the first three phases of the branch and bound are identical. The costs and values of the objective function are different. Also, if a policy

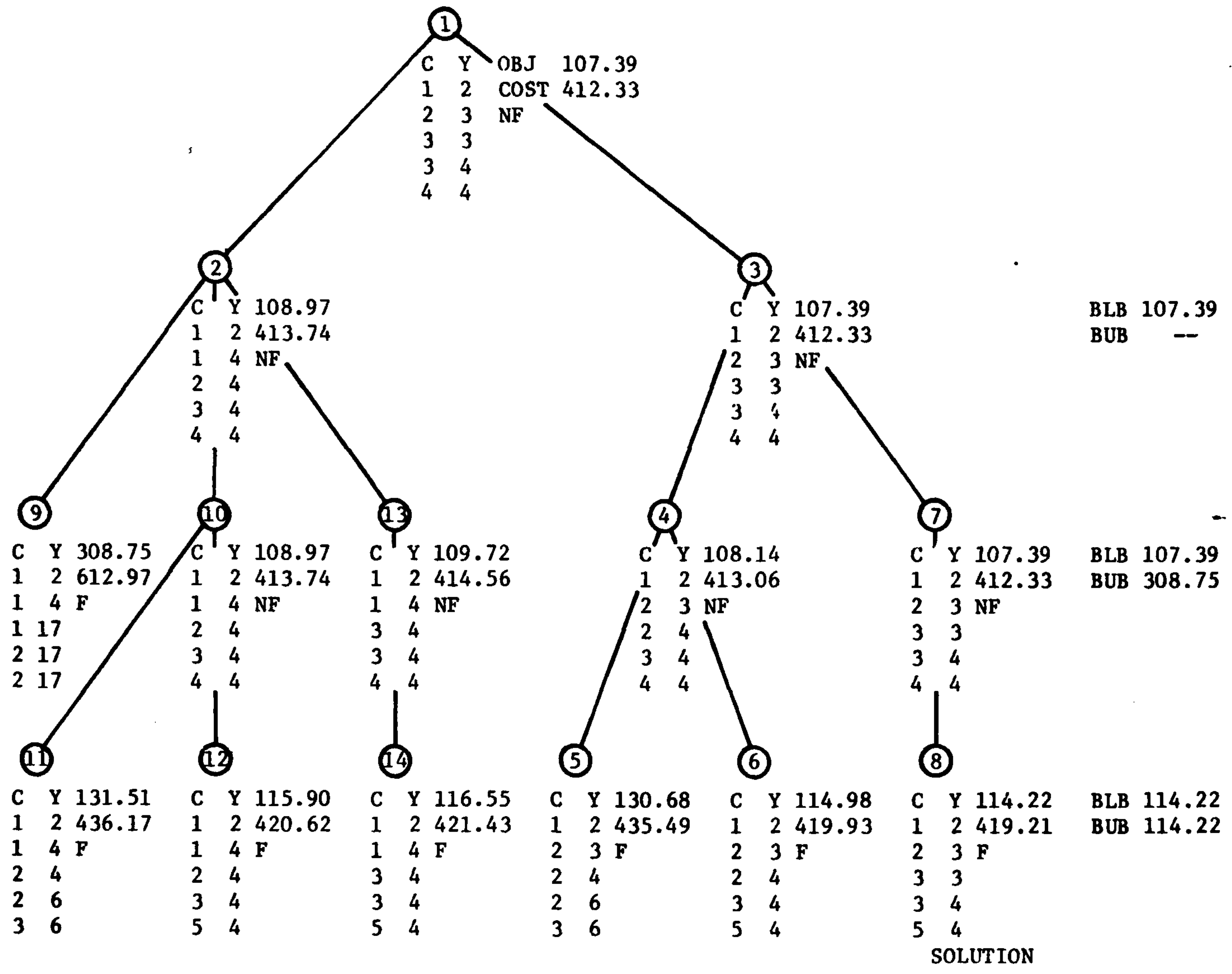


Fig. IV-2. Extended branch and bound tree of Problem B.

```
①
C  Y  OBJECTIVE  96.57
1  2  COST      403.74
1  4  F
2  4
3  4
3  5
```

Fig. IV-3. Branch and bound tree for Problem C.

is not feasible, it is not feasible for all three problems (A, B, and C). Problems A and B differ only in the last phase of the branch and bound structure. In Problem A, because of the order in which the branching was taken, only ten problems were solved. In fact, since a feasible solution was found at Problem 7 whose value of the objective function was equal to the BLB, only seven problems needed to be solved. For Problem B, since no feasible solution had the value of the objective function equal to the BLB of the previous phase, all branches had to be evaluated.

By looking at the three problems (A, B, and C), we see that it is not obvious how efficient the branch and bound procedures may be. For Problem C it was necessary only to solve one problem. For Problem B all of the branches had to be calculated. Problem A could be solved with only about half of the branches evaluated. While it is impossible to predict exactly, it is suspected that the closer the bounds on $\bar{\lambda}_i$, i.e., $\bar{\lambda}_{i-1}(M_{i-1} / M_i)(\lambda_{i-1} - \bar{\lambda}_{i-1}) \approx 0$, for $M_i \geq M_{i-1}$, or $\bar{\lambda}_{i-1}(\lambda_{i-1} - \bar{\lambda}_{i-1}) \approx 0$ for $M_i < M_{i-1}$ [Equation (III-1)], the earlier the branch and bound procedure will obtain the solution.

The primary motivation for this dissertation is based upon the work given in Reference [1]. In that paper a specific gas turbine problem was solved. The input parameters associated with this problem are given in Table IV-3. This is an eleven time-period problem; all costs are given in $1,000 units.

TABLE IV-3

INPUT TO GAS TURBINE PROBLEM

M_i	λ_i	$1/\mu_i$
10	.00147186	65.0000
28	.00152455	62.5000
50	.00136767	60.0000
82	.00099101	57.5000
121	.00082569	55.0000
158	.00071831	55.0000
182	.00063699	55.0000
208	.00060941	55.0000
229	.00061725	55.0000
251	.00062297	55.0000
256	.00062015	55.0000

Costs

C_1	C_2	C_3	C_4
132.00	822.00	49.00	1975.00
132.00	945.00	49.00	2760.00
132.00	1087.00	37.00	3840.00
132.00	1174.00	40.00	3950.00
132.00	1268.00	42.00	2800.00
132.00	1369.00	44.00	1100.00
132.00	1369.00	44.00	350.00
132.00	1369.00	44.00	350.00
132.00	1369.00	44.00	350.00
132.00	1369.00	44.00	350.00
132.00	1369.00	44.00	350.00

Discounted Costs: Factor=.1

C_1	C_2	C_3	C_4
132.00	822.00	49.00	1975.00
120.00	859.09	44.55	2509.09
109.09	898.35	31.24	3173.55
99.17	882.04	30.05	2967.69
90.16	866.06	28.69	1912.44
81.96	850.04	27.32	683.01
74.51	772.77	24.84	197.57
67.74	702.51	22.58	179.61
61.58	638.65	20.53	163.28
55.98	580.59	18.66	148.43
50.89	527.81	16.96	134.94

Availability=.9

Using the exact solution computer program, the solution at the top of Table IV-4 was obtained. This is compared, in this table, to the solution generated using the heuristic approach of Reference [1]. We see that the exact solution is slightly more than $300,000 less costly. The major difference between the optimal policies is that the exact solution buys spares in an earlier time period, when they are cheaper, rather than waiting until they are needed in a later time period. In addition, in the heuristic solution, repair channels are purchased, given up, and then repurchased (time periods 8, 9, 10).

The procedure itemized in this paper is not only exact but also efficient. For the gas turbine problem only one problem had to be solved. There was no branching necessary. A total of only 66 seconds of Hewlett Packard 3000 CPU time and 6 minutes of terminal time, including input of the data and output of the results, was required to solve the problem. This procedure has been tested on only a limited number of problems, but it is expected that it will be significantly more efficient than the heuristic procedure where more interaction and a greater number of iterations are required. How this computational efficiency can be extended to other problems will be discussed in the next chapter.

TABLE IV-4

SOLUTION TO GAS TURBINE PROBLEM

COST = 38827.16

I	c	y	$\overline{R}$
1	2	8	5.371
2	4	8	15.337
3	8	8	26.426
4	10	10	37.197
5	10	12	45.492
6	12	13	51.798
7	12	14	53.967
8	12	14	56.266
9	13	14	58.288
10	15	14	61.583
11	15	14	61.600

Exact Solution

c	y	$\overline{R}$	Present Worth
3	3	5.3	5097.61
5	6	15.3	11105.17
8	8	26.4	17226.10
10	10	37.2	23273.15
12	11	45.5	27537.26
15	12	51.8	30731.45
13	13	53.9	33041.68
14	13	56.3	34559.60
13	14	58.3	36557.30
15	14	61.6	37966.66
15	14	61.6	39146.38

Heuristic Solution

CHAPTER V

SUMMARY, PROBLEM EXTENSIONS, AND CONCLUSIONS

V.1 Summary

A methodology for attaining an exact solution to the problem of minimizing the cost of a multi-year spares provisioning problem has been developed in this dissertation. The problem solved is a nonlinear integer programming problem, and was previously solved using a heuristic algorithm. There was no assurance that the heuristic solution was optimal or even near optimal. However, comparison of results of this effort and the heuristic indicate that the heuristic, in the cases tested, gives results very close to optimal.

There are a number of difficulties associated with this problem, beyond those normally associated with integer programming problems. The forms of the equations of state are dependent not only on the state variables, but also on the relative values of these variables as well as other parameters of the problem. In addition, a term, $\overline{\lambda}$, which is normally a parameter of the problem, is a function of the state variables. To resolve these problems, an amalgamation of techniques has been used to obtain the solution. Branch and bound is used to reparameterize $\overline{\lambda}$. Enumeration and a construct of "corners" are used to specify the feasible region of the subproblems. Finally, dynamic programming is used to obtain a solution. This dissertation demonstrates that this approach gives an exact optimal solution and that this solution is generated in an efficient manner.

This effort was motivated by the need to obtain an exact solution to a specific problem, the minimum cost, multi-year policy for purchasing spares and repair channels for a gas turbine maintenance system. There is no reason why the applicability of the solution technique is limited to the particular gas turbine problem. The underlying stochastic model, the machine repair problem with spares, is the foundation for analysis of many systems problems. This chapter discusses the types of problem extensions that can be handled with minor modifications to the solution framework. Also discussed is the potential applicability of some of the concepts developed in obtaining the solution techniques.

V.2 Problem Extension

There are three basic factors that determine the nature of the solution to this problem. They are:

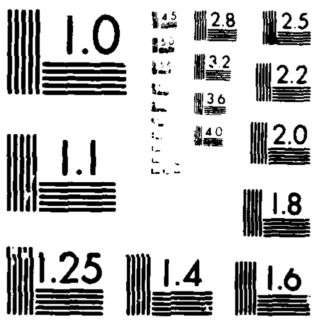
1. a linear objective function;
2. a stochastic availability constraint based upon the equations of state associated with the classical machine repair problem with spares;
3. a key parameter in determining the equations of state, $\overline{\lambda}_i$, the mean failure rate, which is an implicit function of the decision variables.

Each of these factors had significant impact on the choice of techniques required for an exact solution. The linearity of the objective function forces the solution to lie on the convex hull of the feasible region. This allows an enumeration of possible solution sets because the region can be described by a finite number of points. The use of the solution to the machine repair problem provided a functional dependence on the state variables that could be enumerated. Finally, the dependence of $\overline{\lambda}_i$ on the state variables requires the use of the branch and bound procedure to approximate

AD-A084 035 GEORGE WASHINGTON UNIV WASHINGTON D C PROGRAM IN LOG--ETC F/G 15/5
AN OPTIMIZATION TECHNICAL FOR A MULTITIME PERIOD SPARES PROVISI--ETC(U)
FEB 80 H K RAPPOPORT
UNCLASSIFIED SERIAL-T-408 NL

2 OF 2
AD
A084035

END
DATE
FILMED
6-80
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

successively the problem in a manner that assures an exact solution.

Dynamic programming is not necessary for obtaining a solution. Once the convex hull of the feasible region has been enumerated, it could be described by linear inequalities, allowing solution through linear programming techniques. However, dynamic programming provides computational efficiency and latitude in modification of the types of problems that can be solved.

The first possibility is modification of the objective function. Its linearity was used only to guarantee that the problem's solution is on the convex hull. Since any objective function that is isotonic increasing [a function is isotonic if, for vectors a and b where $a \geq b$, then $f(a) \geq f(b)$] in the state variables will also guarantee this, such extended problems can be solved through the procedure described.

Continuing work on the gas turbine problem has resulted in a modification in the objective function. The new objective function is

$$\sum_{i=1} d^i \Big[P_{c,i}(c_i - c_{i-1})^+ + P_{y,i}(y_i - y_{i-1})^+ + S_{c,i}(c_i - c_{i-1})^- + S_{y,i}(y_i - y_{i-1})^- + C_{o,i}\, c_i + C_{I,i} \sum_{n=0}^{y_i} (y_i - n) P_n + C_{R,i}\, \overline{R}_i + C_{F,i}(\lambda_i) \Big] ,$$

where

$P_{c,i}$ = purchase cost of a server, year i

$P_{y,i}$ = purchase cost of a spare, year i

$S_{c,i}$ = salvage cost of a server, year i

$S_{y,i}$ = salvage cost of a spare, year i

$C_{R,i}$ = repair cost per unit, year i

$C_{o,i}$ = operating cost of a server, year i

$C_{I,i}$ = inventory carrying cost, year i

$C_{F,i}$ = cost of CIP, year i

$(a)^+ = a$ if $a > 0$ and 0 if $a \leq 0$;

$(a)^- = a$ if $a < 0$ and 0 if $a \geq 0$.

It has not been determined if this objective function is isotonic. It is expected that an objective function containing only the first five terms can be handled using the technique of the exact solution. While such an objective function is not necessarily isotonic, it should be possible to enumerate all potential interior solution points. These states can be added to the state space of the dynamic program. Whether the entire objective function can be handled depends on the nature of the repair, inventory and CIP costs, which requires further analysis. If such an objective function is used, an additional modification to the problem would have to be made. The constraints $c_i \geq c_{i-1}$ and $y_i \geq y_{i-1}$ are no longer valid. The generation of the feasible space would have to be changed. It is expected, however, that these changes could be made within the general framework of the solution with little modification of the computer program.

Another extension of the original gas turbine problem is a modification of the definition of the availability constraint. Instead of requiring a minimum spare availability at the time of failure, the constraint is placed on the overall minimum fleet availability over the total time period. This is equivalent to replacing the constraints

$$\sum_{n=0}^{y-1} q_{n,i} \geq a$$

by

$$\sum_{n=0}^{y} p_{n,i} \geq a ,$$

where

$q_n \equiv \Pr\{n \text{ in system} | \text{arrival is about to occur}\}$,

$p_n \equiv \Pr\{n \text{ in system}\}$,

a is the availability required.

This new constraint is computationally simpler to work with and should satisfy the important property that if the availability constraint is satisfied for a pair (c,y) then it is satisfied for a pair (c',y') , where $c' \geqq c$ and $y' \geqq y$.

These have been explicit modifications to the problem that have application to ongoing work. The general nature of the problem with spares arises in industries as diverse as utilities to automobile repair. While each problem has its own features, the generality of the approach should allow solution of a number of different types of problems.

Another aspect of this dissertation, that may have use beyond this immediate application, is the concept of "corners." In this dissertation very simple relationships between point sets were handled in a computationally efficient manner. It may be possible to handle more complex relationships with the same efficiency. This technique simplifies the enumeration of states associated with the integer programs. Further exploration of this concept may expedite the solution of other types of integer programs.

V.3 Conclusion

In an attempt to obtain an exact solution to a nonlinear integer programming problem, a diverse group of techniques was brought together into a single package that is easy to use and computationally efficient. It is hoped that these techniques can find wider application in the general area of problems associated with the classical problem of machine repair with spares.

APPENDIX A

LISTING OF COMPUTER PROGRAM

```
C   THIS PROGRAM DETERMINES THE OPTIMAL POLICY FOR A SPARES
C   PROVISIONING PROBLEM.  A BRANCH AND BOUND APPROACH IS
C   USED.  THEREFORE SUBPROBLEMS MAY HAVE TO BE INPUT SEQUENT-
C   IALLY.  THIS APPROACH ALSO INCORPORATES ENUMERATION
C   TECHNIQUES AND DYNAMIC PROGRAMING.
C*************************************************************
C  INPUT:
C   NT=NUMBEROF TIME PERIODS
C   M(I)=NUMBER OF MACHINES,TIME PERIOD I
C   X(I)=LAMBDA (FAILURE RATE) FOR TIME PERIOD I
C   X1(I)=TIME BETWEEN REPAIRS (1/MU) FOR TIME PERIOD I
C   SC(I,J)=COSTS FOR TIME PERIOD I.J=1,2,3,4
C             ASSOCIATED WITH C1,C2,C3,C4,OF THE OBJECTIVE
C             FUNCTION RESPECTIVELY
C   DIS=DISCOUNT FACTOR
C   AV=AVAILABILITY REQUIREMENT
C   IP+PRINT SWITCH;=1 FULL PRINT; OTHERWISE PRINT SOLUTION
C                      OUTPUT ONLY
C   IBB=BRANCHING SWITCH;=0 STOP;OTHERWISE THE NUMBER OF
C                          THE BRANCHING TIME PERIOD
C   IX(K,1,1)=NUMBER OF REPAIR CHANNELS SPECIFIED FOR TIME
C             PERIOD K FOR BRANCHING PROBLEM.
C             1<K<IBB
C   IX(K,1,2)=NUMBER OF SPARES SPECIFIED FOR TIME PERIOD
C             K FOR BRANCH PROBLEM.
C             1<K<IBB
C   R(K)=MEAN NUMBER OF MACHINES REPAIRED IN TIME PERIOD
C        K FOR BRANCHING PROBLEM.
C        1<K<IBB
C*****************************************
C  OUTPUT
C   SC(I,J)=C1,C2,C3,C4 FOR J+1,2,3,4, RESPECTIVELY, FOR TIME
C             PERIOD I
C           =DISCOUNTED COST C1,C2,C3,C4
C           =DIFFERENCE IN COST BETWEEN TIME PERIODS I AND
C           I+1, FOR C1 AND C2, J=1,2,RESPECTIVELY
C**SC(I,J) IS USED FOR THREE DIFFERENT COSTS TO  REDUCE
C**MEMMORY REQUIREMENTS OF THE COMPUTER PROGRAM**
C   IX(I,J,K)=CORNERS OF THE DOMINATED REGIONS(SET OF
C             MINIMUM PAIRS(C,Y) SATISFYING THE CONSTRAINTS)
C             I=1 TO NT, NUMBER OF TIME PERIODS
C             J=NUMBER OF PAIRS IN THE SET
C             K=1;NUMBER OF REPAIR CHANNELS
C               2;NUMBER OF SPARES
C   ID(I)=NUMBER OF PAIRS IN SET FOR TIME PERIOD I; J<ID(I)
C   CT=TOTAL COST OF THE SOLUTION INCLUDING C1,C2,C3,C4
C   CS=COST OF THE OBJECTIVE FUNCTION,C1 AND C2 TERMS ONLY
C   I=TIME PERIOD
C   JS(I,1)=NUMBER  OF REPAIR CHANNELS AVAILABLE IN TIME PERIO
```

D I

```
C           OF THE SOLUTION TO THE SUBPROBLEM
C    JS(I,2)=NUMBER OF SPARES AVAILABLE IN TIME PERIOD I OF THE
C         SOLUTION TO THE SUBPROBLEM
C    R(I)=MEAN NUMBER OF MACHINES REPAIRED IN TIME PERIOD I
C          OF THE SOLUTION TO THE SUBPROBLEM
C    Z(I)=LAMBDA BAR (AVERAGE FAILURE RATE) FOR THE TIME PERIOD
C          I ASSOCIATED WITH THE SOLUTION TO THE SUBPROBLEM
C    IBD=BRANCHING TIME PERIOD
C    ICY=NUMBER OF REPAIR CHANNELS
C    IYY=NUMBER OF SPARES
C    RT=MEAN NUMBER OF MACHINES REPAIRED
C**************************************************************
C
C
C   THIS IS THE MAIN PROGRAM. IT ACCEPTS INPUT, WRITES OUTPUT
C   ,AND GOVERNS THE FLOW OF THE SOLUTION.  IT AUTOMATICALLY
C   GENERATES ONE SET OF BRANCHES IF REQUIRED.
      DIMENSION M(15),X(15),X1(15),R(15),Z(15),JS(15,2),
     1ID(15),SC(15,4),IX(15,25,2),RS(25)
      WRITE(6,910)
      ACCEPT NT
      WRITE(6,911)
      DO 91 I=1,NT
   91 ACCEPT M(I),X(I),X1(I)
      Z(1)=X(1)
      WRITE(6,902) (M(I),X(I),X1(I),I=1,NT)
      WRITE(6,912)
      DO 92 I=1,NT
   92 ACCEPT(SC(I,J),J=1,4)
      WRITE(6,917)((SC(I,J),J=1,4),I=1,NT)
      WRITE(6,913)
      ACCEPT DIS,AV
      WRITE(6,918) DIS,AV
      DD=1
      DO 1 I=2,NT
      DD=DD/(DIS+1.)
      DO 1 J=1,4
    1 SC(I,J)=SC(I,J)*DD
      WRITE(6,917) ((SC(I,K),K=1,4),I=1,NT)
      R(1)=0
C   CALCULATE COST DIFFERENCES-USED IN MODIFIED OBJECTIVE
C   FUNCTION
      DO 2 I=2,NT
      J=I-1
      SC(J,1)=SC(J,1)-SC(I,1)
    2 SC(J,2)=SC(J,2)-SC(I,2)
      WRITE(6,914)
      ACCEPT IP
      IF(IP.EQ.1) WRITE(6,906) ((SC(I,K),K=1,4),I=1,NT)
      BUB=100000000
      IBB=1
    3 LS=1
C  BRANCH AND BOUND LOOP
      DO 250 IN=1,2
      DO 200 N=1,LS
      IR=0
      DO 100 K=1,NT
      IF(K  .LE.1) GO TO 4
      J=K-1
      I=K
      IF(M(I).LT.M(J)) I=J
C  CALCULATE LAMBDA CAP
      Z(K)=((M(I)-M(J))*X(K)+R(J)*X(J)+(M(J)-R(J))*Z(J))/M(I)
    4 CONTINUE
      IF(K.LT.IBB) GO TO 100
      R(K)=0
      IF(Z(K).GE.X(K)) R(K)=365.*Z(K)*M(K)
C  CALCULATE RHO
      XM=Z(K)*M(K)*X1(K)
      L=0
      IYO=0
      ICO=0
      ICY=XM-1
      IF(ICY.LT.0)ICY=0
    5 ICY=ICY+1
      IYY=ICY
C CALCULATE MINIMUM (C,Y) PAIRS
   10 CALL QUE(XM,M(K),AV,ICY,IYY,IR,X1(K),RT)
      IF(IP.EQ.1) WRITE(6,901) ICY,IYY
      IF(ICY.GT. IYY+M(K)) GO TO 70
C DETERMINE IF C AND Y ARE BOTH NEW
      IF((IYY-IYO)*(ICY-ICO).EQ.0) GO TO 60
      L=L+1
       ICO=ICY
      IYO=IYY
C IX IS A CORNER POINT
      IX(K,L,1)=ICY
      IX(K,L,2)=IYY
   60 CONTINUE
C  TEST IF Y>C
       IF (IYY.GT.ICY) GO TO 5
C  TEST IF ALL CORNER POINTS HAVE BEEN GENERATED FOR THIS
C  TIME PERIOD
      IF(IYY.EQ.1) GO TO 70
       IYY=IYY-1
       GO TO 10
   70 ID(K)=L
      IF(IP.EQ.1) WRITE(6,905) (K,IX(K,I,1),IX(K,I,2),I=1,L)
  100 CONTINUE
```

```
      WRITE(6,903) (Z(I),I=1,NT)
C  MERGE DOMINATED REGIONS
      CALL UNION(NT,IX,ID)
      IF(IP.EQ.1)WRITE(6,905)((K,(IX(K,I,J),J=1,2),I=1,ID(K)),
K=1,NT)
C DO DYNAMIC PROGRAMMING
      CALL DP(NT,IX,ID,SC,JS,CS,IP)
      CT=CS
C  TEST IF SOLUTION IS FEASIBLE FOR ORIGINAL PROBLEM
      X0=0.
      R0=0.
      M0=0
      Z0=0.
      IRS=0
      IR=1
      DO 150 I=1,NT
C   CALCULATE LAMBDA BAR
      Z(I)=((M(I)-M0)*X(I)+R0*X0+(M0-R0)*Z0)/M(I)
      XM=Z(I)*M(I)*X1(I)
      ICY=JS(I,1)
      IYY=JS(I,2)
      IR=1
C TEST IF AVAILABILITY CONSTRAINT IS SATISFIED
      CALL QUE(XM,M(I),AV,ICY,IYY,IR,X1(I),RT)
      R(I)=365.*RT
      X0=X(I)
      R0=R(I)
      M0=M(I)
      Z0=Z(I)
      IF(IR.LT.0) IRS=1
C  CALCULATE TOTAL COST
  150 CT=CT+R(I)*SC(I,3)+SC(I,4)
      WRITE(6,908) CT,CS,(I,JS(I,1),JS(I,2),R(I),I=1,NT)
      WRITE(6,903) (Z(I),I=1,NT)
C  TEST IF FEASIBLE. IF INFEASIBLE BRANCHING IS REQUIRED
      IF(IRS.EQ.0) WRITE(6,916)
C  TEST IF THE SOLUTION IS A BEST UPPER BOUND
      IF(CT.GT.BUB) GO TO 170
       IF(IRS.NE.0) GO TO 155
C  HAS BRANCHING ALREADY OCCURED
      IF(IBB.EQ.1) STOP
      BUB=CT
      GO TO 170
  155 CONTINUE
C  BRANCH
```

```
      IF(IBB.EQ.1) IBB=2
C  DETERMINE TIME PERIOD FOR BRANCHING
      IBD=IBB+1-IN
      WRITE(6,909) IBD
      L=ID(IBB)
C  CALCULATE RHO
      XM=Z(IBB)*M(IBB)*X1(IBB)
C  DETERMINE LAMBDA BAR FOR THE TIME PERIOD PRIOR TO THE
C  BRANCHING TIME PERIOD
      DO 160 I=1,L
      ICY=IX(IBB,I,1)
      IYY=IX(IBB,I,2)
      IR=1
C  CALCULATE LAMBDA BAR
      CALL QUE(XM,M(IBB),AV,ICY,IYY,IR,X1(IBB),RT)
      RT=RT*365
      IF(IN.EQ.1) RS(I)=RT
  160 WRITE(6,915) ICY,IYY,RT
  170 LN=IN+N-1
      IF(LN.GT.LS) GO TO 200
C  DETERMINE BRANCHES
      IX(IBD,1,1)=IX(IBD,LN,1)
      IX(IBD,1,2)=IX(IBD,LN,2)
      R(IBD)=RS(LN)
  200 CONTINUE
C  TEST IF THIS BRANCH IS GREATER THAN THE BUB
      IF(CT.GE.BUB) GO TO 300
      ID(IBB)=1
      IBB=IBB+1
  250 LS=L
  300 WRITE(6,904)
C  INPUT NEW BRANCH
      ACCEPT IBB
      IF(IBB.LE.0) STOP
      IF(IBB.LT.3) GO TO 4
      J=IBB-1
      DO 35 K=2,J
      WRITE(6,907) K
C  INPUT POLICY ASSOCIATED WITH NEW BRANCH
      ACCEPT IX(K,1,1),IX(K,1,2),R(K)
   35 ID(K)=1
      GO TO 3
  901 FORMAT(2I5,F10.3)
  902 FORMAT('     M     LAMBDA     1/MU'/11(I6,F11.8,F8.4/))
  903 FORMAT(/// LBAR=',6F10.8/6X,6F10.8//)
  904 FORMAT(/// INPUT BRANCHING TIME PERIOD')
  905 FORMAT(' K=',I5,' C=',I5,' Y=',I5)
  906 FORMAT( 4F10.2)
  907 FORMAT(' INPUT C,Y AND RBAR  FOR TIME PERIOD ',I2)
  908 FORMAT(/3X,'COST=',F10.2,8X,'OBJECTIVE=',F10.2/
          12X,'I',4X,'C',4X,'Y',7X,'R'/(3I5,F10.3))
      909 FORMAT(' BRANCHING ON TIME PERIOD ',I3)
      910 FORMAT(5X,'ENTER NUMBER OF TIME PERIODS'/)
      911 FORMAT(5X,'ENTER M,X,1/MU'/)
      912 FORMAT(5X,'ENTER COSTS'/)
      913 FORMAT(5X,'ENTER DISCOUNT FACTOR AND AVAILABILITY REQUIR
EMENT'/)
      914 FORMAT(5X,'IF DIAGNOSTIC PRINT DESIRED ENTER 1')
      915 FORMAT(' C=',I4,' Y=',I4,' RBAR=',F12.8)
      916 FORMAT(' FEASIBLE SOLUTION'/)
      917 FORMAT(4X,'C1',8X,'C2',8X,'C3',8X,'C4'/15(4F10.2))
      918 FORMAT(' DISCOUNT FACTOR=',F6.4,' AVAILABILITY=',F6.4)
          END
          SUBROUTINE UNION(NT,IX,ID)
C  THIS SUBROUTINE CALCULATES THE CORNERS GENERATED BY
C  TAKING THE INTERSECTION OF TWO DOMINATED REGIONS
C         NT=THE NUMBER OF TIME PERIODS
C         IX(I,J,K)=THE (C,Y) PAIR DEFINING THE CORNERS
C                   I=THE TIME PERIOD
C                   J=THE CORNER INDEX
C                   K=1 FOR C;2 FOR Y
C         ID(I)=THE NUMBER OF CORNERS NEEDED TO GENERATE THE
C                DOMINATED REGION FOR TIME PERIOD I: J<=ID(I)
          DIMENSION IX(15,25,2),ID(15),K(2),J(3),LL(25,2)
          NF=NT-1
C  FIND INTERSECTION OF DOMINATED REGIONS
          DO 100 N=1,NF
          N1=N+1
          K(1)=ID(N)
          K(2)=ID(N1)
          J(1)=1
          J(2)=1
          J(3)=0
          L=1
C  DETERMINE MAXIMUM VALUE OF C
          IF(IX(N,J(1),1).LT.IX(N1,J(2),1)) L=2
          M=N+L-1
          M1=N1+1-L
C  SAVE MAXIMUM VALUE
        5 IXT=IX(M,J(L),1)
          L1=3-L
       10 J(L1)=J(L1)+1
          IF(J(L1).GT.K(L1)) GO TO 15
C  TEST IF DOMINATED BY NEXT POINT
          IF(IX(M1,J(L1),1).LE.IXT) GO TO 10
       15 J(L1)=J(L1)-1
          J(3)=J(3)+1
          LL(J(3),1)=IXT
          IF(IX(M,J(L),2).LT.IX(M1,J(L1 ),2)) L=L1
          M=N+L-1
      M1=N1+1-L
      LL(J(3),2)=IX(M,J(L),2)
      L1=3-L
      IXT=IX(M1,J(L1),2)
   20 J(L)=J(L)+1
      IF(J(L).GT.K(L)) GO TO 30
      IF(IX(M,J(L),2).LT.IXT) GO TO 5
      J(3)=J(3)+1
C NEW CORNER
      LL(J(3),1)=IX(M,J(L),1)
      LL(J(3),2)=IX(M,J(L),2)
      GO TO 20
   30 CONTINUE
      ID(N1)=J(3)
      DO 40 L=1,J(3)
      IX(N1,L,1)=LL(L,1)
   40 IX(N1,L,2)=LL(L,2)
  100 CONTINUE
      RETURN
      END
      SUBROUTINE DP(NT,IX,ID,SC,JS,CS,IP)
C   THIS SUBROUTINE FINDS THE SOLUTION TO A SUBPROBLEM
C   USING DYNAMIC PROGRAMMING
C       NT=THE NUMBER OF TIME PERIODS
C       IX(I,J,K)=SET OF POSSIBLE SOLUTION POINTS- CORNERS
C                 I=THE TIME PERIOD
C                 J=THE PAIR INDEX
C                 K=1 FOR C;2 FOR Y
C       ID(I)=THE SIZE OF IX SET FOR TIME PERIOD I:J<=ID(I)
C       SC(I,L)=COSTS IN THE OBJECTIVE FUNCTION
C               I=THE TIME PERIOD
C               L=1 TO 4;ONLY 1 AND 2 ARE USED IN THE
C                 OBJECTIVE FUNCTION
C       JS(I,K)=SOLUTION POLICY
C               I=TIME PERIOD
C               K=1 FOR C;2 FOR Y
C       JX(I,K)=THE SOLUTION POLICY ASSUMING A PARTICULAR
C               POLICY FOR THIS LAST TIME PERIOD (I=NT)
C               I=THE TIME PERIOD
C               K=1 FOR C;2 FOR Y
      DIMENSION IX(15,25,2),ID(15),SC(15,4),JS(15,2),JX(15,2)
      CS=100000000.
      KF=ID(NT)
      IN=NT-1
C   FIND A SOLUTION FOR EACH POINT IN THE LAST TIME PERIOD(NT)
      DO 1000 K=1,KF
      I1=IX(NT,K,1)
      I2=IX(NT,K,2)
C   DETERMINE VALUE OF OBJECTIVE FUNCTION FOR THE LAST
C   TIME PERIOD
```

```
          C=I1*SC(NT,1)+I2*SC(NT,2)
          JX(NT,1)=I1
          JX(NT,2)=I2
C   DETERMINE BEST POLICY FOR  SUBSEQUENT STAGES (EARLIER
C   TIME PERIODS)
          DO 100 I=1,IN
          NI=NT-I
          L=0
C   DETERMINE OPTIMAL Y
      10 L=L+1
C   TEST IF  <= CONSTRAINT IS SATISFIED
          IF(IX(NI,L,2).GT.I2) GO TO 10
C   CALCULATE CONTRIBUTION TO OBJECTIVE FUNCTION
          CT=I2*SC(NI,2)+IX(NI,L,1)*SC(NI,1)
          JC=IX(NI,L,1)
          JY=I2
          L=L-1
      20 L=L+1
C   DETERMINE OPTIMAL Y
          IF(L.GT.ID(NI)) GO TO 50
C   TEST FOR END OF STATES
          IF(IX(NI,L,1).GT.I1) GO TO50
C   CALCULATE CONTRIBUTION TO OBJECTIVE FUNCTION
          CI=IX(NI,L,1)*SC(NI,1)+IX(NI,L,2)*SC(NI,2)
          IF(CI.GT.CT) GO TO 20
          CT=CI
          JC=IX(NI,L,1)
          JY=IX(NI,L,2)
          GO TO 20
C   TEST FOR BEST POLICY
      50 CI=SC(NI,1)*JX(NI+1,1)+SC(NI,2)*JX(NI+1,2)
          IF(CI.GT.CT) GO TO 60
          CT=CI
C   SAVE POLICY
          JC=JX(NI+1,1)
          JY=JX(NI+1,2)
C   SAVE OPTIMAL STATES
      60 JX(NI,1)=JC
          JX(NI,2)=JY
          C=C+CT
          I1=JC
          I2=JY
     100 CONTINUE
C   TEST EACH SOLUTION ASSOCIATED WITH A STATE OF THE LAST
C   TIME PERIOD TO DETERMINE THE MINIMUM OBJECTIVE FUNCTION
          IF(IP.EQ.1) WRITE(6,901) (C,JX(I,1),JX(I,2),I=1,NT)
     901 FORMAT (F10.2,2I10)
          IF(C.GT.CS) GO TO 1000
          CS=C
          DO 150 I=1,NT
```

```
C  SAVE SOLUTION
        JS(I,1)=JX(I,1)
        JS(I,2)=JX(I,2)
    150 CONTINUE
   1000 CONTINUE
        RETURN
        END
         SUBROUTINE QUE(RHO,M,AV,IC,IY,IR,U,RBAR)
C  THIS SUBROUTINE CALCULATES THE AVAILABILITY CONSTRAINT
C  AND R BAR (MEAN NUMBER OF MACHINES REPAIRED PER DAY)
C  IT DETERMINES THE MINIMUM VALUES SATISFYING THE AVAILABILIT
Y
C  CONSTRAINT, OR IF A (C,Y) PAIR IS SPECIFIED, DETERMINES
C  IF IT SATISFIES THE CONSTRAINT.
C      RHO=RHO (LAMBDA BAR X M/MU)
C      M=THE NUMBER OF MACHINES
C      AV=THE AVAILABILITY REQUIREMENT
C      IC=C
C      IY=Y
C      IR=SWITCH:=1 TEST IF AVAILABILITY CONSTRAINT IS
C                 SATISFIED FOR A SPECIFIED (C,Y) PAIR
C                  NOT 1 GENERATE (C,Y) PAIR
C      U=MU (REPAIR RATE)
C      RBAR=RBAR( MEAN NUMBER OF MACHINES REPAIRED)
C*****CALCULATE A
        A=AV/(1-AV)
C  TEST IF C>Y
        IF(IC.GT.IY) GOTO 600
C  TEST IF C= RHO
    10  RC=RHO/IC
        IF(RC.NE.1)RR=1/(1-RC)
        S2=0
        T2=1
C CALCULATE S2
        DO 100 J=1,M
        S2=S2+T2
        IF(T2.LT..00000001) T2=0
   100  T2=T2*RHO/(M*IC)*(M-J)
        IF(IR.EQ.1) GO TO 150
        IF(RC.LE.1) GOTO 150
C  TEST IF AVAILABILITY CONSTRAIN CAN BE SATISFIED FOR Y>C
        IF(A*S2.LE.-RR) GOTO 150
        IC=IC+1
        GO TO 10
   150  S1=0
C  CALCULATE S1
         T1=1
        DO 200 I=1,IC
        S1=S1+T1
        IF(T1.LT..00000001) T1=0
```

```
C  CALCULATE Y
  200  T1=T1*RHO/I
       IF(IR.EQ.1) GOTO 400
       TF =A*S2-S1/T1
       IF(RC.NE.1) TF=ALOG((S1/T1+RR)/(A*S2+RR))/ALOG(RC)
       IY=IC+TF+.999999
       IF(TF.LT.-.0001) GO TO 500
       GO TO 1000
  400  T3=RC**(IY-IC)
C  TEST IF (C,Y) SATISFY AVAILABILITY CONSTRAINT
       S1=S1+T1*(1.-T3)*RR
       T1=T1*T3
        S2=T1*S2
       GO TO 2000
C  Y<C
  500  IY=IC
  600  S1=0.
       T1=1.
C  CALCULATE AVAILABILITY CONSTRAINT
       DO 700 I=1,IY
       S1=S1+T1
       IF(T1.LT..00000001) T1=0
  700  T1=T1*RHO/I
        S2=0.
       T2=1.
       DO 800 J=1,IC-IY
       S2=S2+T2
       IF(T2.LT..00000001) T2=0
  800  T2=T2/M*RHO/(IY+J)*(M-J)
       DO 900 J=IC-IY+1,M
       S2=S2+T2
       IF(T2.LT..00000001) T2=0
  900  T2=T2/IC*RHO/M*(M-J)
       S2=S2*T1
C  ALREADY KNOW (C,Y). WANT TO TEST IF AVAILABILITY
C  CONSTRAINT IS SATISFIED AND WANT TO CALCULATE RBAR
       IF(IR.EQ.1) GO TO 2000
C  TEST IF AVAILABILITY CONSTRAINT IS SATISFIED
       IF (S1.GE.A*S2) GO TO 1000
C  INCREASE  C
       IC=IC+1
C  TEST IF C>Y+M
       IF( IC.LE.IY+M) GO TO 600
 1000  RETURN
C  CALCULATE RBAR
 2000  RBAR=RHO/U*(S1+S2)/(S1+RHO/IC*S2+T1)
C  AVAILABILITY CONSTRAINT NOT SATISFIED
       IF (S1.LT.A*S2) IR=-1
       RETURN
       END
```

APPENDIX B

ALGEBRAIC REFORMULATIONS OF THE AVAILABILITY CONSTRAINT

This appendix will give the reconfiguration of the availability constraint in a more convenient form. The availability constraint is given by:

$$\sum_{n=0}^{y-1} q_n \geq 0.90 \ .$$

This can be generalized for any availability to

$$\sum_{n=0}^{y-1} q_n \geq a \ ,$$

where a is the availability.

Using the expression (II-4) for q_n (and noting that $n \leq y-1$) , we have

$$\frac{\sum_{n=0}^{y-1} Mp_n}{M - \sum_{\ell=y}^{y+M} (\ell-y)p_\ell} \geq a \ .$$

This can be rewritten as

$$\frac{\sum_{n=0}^{y-1} p_n}{1 - \sum_{j=0}^{M} \frac{j}{M} p_{j+y}} \geq a \ .$$

Using the property of probability $\sum_{n=0}^{y+M} p_n = 1$, we have

$$\frac{\sum_{n=0}^{y-1} p_n}{\sum_{n=0}^{y-1} p_n + \sum_{n=y}^{y+M} p_n - \sum_{j=0}^{M} \frac{j}{M} p_{j+y}} \geqq a .$$

Changing an index and regrouping,

$$(1-a) \sum_{n=0}^{y-1} p_n \geqq a \left[\sum_{j=0}^{M} \left(\frac{M}{M}\right) p_{j+y} - \sum_{j=0}^{M} \left(\frac{j}{M}\right) p_{j+y} \right] .$$

Finally,

$$\sum_{n=0}^{y-1} p_n \geqq \left(\frac{a}{1-a}\right) \sum_{j=0}^{M-1} \left(\frac{M-j}{M}\right) p_{j+y} . \qquad \text{(B-1)}$$

REFERENCES

[1] GROSS, D., H. D. KAHN, and J. D. MARSH (1977). Queueing models for spares provisioning, *Naval Research Logostics Quarterly 24*, (4), 521-536 (December).

[2] TAYLOR, J. and R. R. P. JACKSON (1954). An application of the birth-death process to the provision of spare machines, *Operational Research Quarterly* 5, 95-108.

[3] BARZILY, Z. and D. GROSS (1979). Transient solutions for repairable item provisioning, Technical Paper Serial T-390, Program in Logistics, The George Washington University (April).

[4] GROSS, D. and J. F. INCE (1978). Spares provisioning for a heterogeneous population, Technical Paper Serial T-376, Program in Logistics, The George Washington University (June).

[5] GROSS, D. and H. D. KAHN (1976). On the machine repair problem with spares under unequal failure rates, Technical Memorandum Serial TM-66451, Program in Logistics, The George Washington University.

[6] GASS, S. I. (1969). *Linear Programming Methods and Applications*. McGraw-Hill Book Company, New York.

[7] LAWLER, E. L. and D. E. WOOD (1966). Branch and bound methods: A survey, *Operations Research 14*, (4), 699-719 (July-August).

[8] MITTEN, L. G. (1970). Branch and bound methods: General formulation and properties, *Operations Research Quarterly 18*, (1), 24-34 (January-February).

[9] HILLIER, F. S. and G. J. LIEBERMAN (1969). *Operations Research*. Holden-Day, Inc., San Francisco.

[10] DANO, S. (1975). *Nonlinear and Dynamic Programming, An Introduction*. Springer-Verlag, New York.

THE GEORGE WASHINGTON UNIVERSITY
Program in Logistics
Distribution List for Technical Papers

The George Washington University
  Office of Sponsored Research
  Library
  Vice President H. F. Bright
  Dean Harold Liebowitz
  Dean Henry Solomon

ONR
  Chief of Naval Research
    (Codes 200, 434)
  Resident Representative

OPNAV
  OP-40
  DCNO, Logistics
  Navy Dept Library
  NAVDATA Automation Cmd
  OP-964

Naval Aviation Integrated Log Support

NARDAC Tech Library

Naval Electronics Lab Library

Naval Facilities Eng Cmd Tech Library

Naval Ordnance Station
  Louisville, Ky.
  Indian Head, Md.

Naval Ordnance Sys Cmd Library

Naval Research Branch Office
  Boston
  Chicago
  New York
  Pasadena
  San Francisco

Naval Ship Eng Center
  Philadelphia, Pa.
  Washington, DC

Naval Ship Res & Dev Center

Naval Sea Systems Command
  PMS 30611
  Tech Library
  Code 073

Naval Supply Systems Command
  Library
  Operations and Inventory Analysis

Naval War College Library
  Newport

BUPERS Tech Library

FMSO

Integrated Sea Lift Study

USN Ammo Depot Earle

USN Postgrad School Monterey
  Library
  Dr Jack R. Borsting
  Prof C. R. Jones

US Marine Corps
  Commandant
  Deputy Chief of Staff, R&D

Marine Corps School Quantico
  Landing Force Dev Ctr
  Logistics Officer
  Commanding Officer
    USS Francis Marion (LPA-249)

Armed Forces Industrial College

Armed Forces Staff College

Army War College Library
  Carlisle Barracks

Army Cmd & Gen Staff College

Army Logistics Mgt Center
  Fort Lee

Commanding Officer, USALDSRA
  New Cumberland Army Depot

Army Inventory Res Ofc
  Philadelphia

Air Force Headquarters
  AFADS-3
  LEXY
  SAF/ALG

Griffiss Air Force Base
  Reliability Analysis Center

Gunter Air Force Base
  AFLMC/XR

Maxwell Air Force Base Library

Wright-Patterson Air Force Base
  Log Command
  Research Sch Log
  AFALD/XR

Defense Documentation Center

National Academy of Sciences
  Maritime Transportation Res Board Library

National Bureau of Standards
  Dr B. H. Colvin
  Dr Joan Rosenblatt

National Science Foundation

National Security Agency

Weapon Systems Evaluation Group

British Navy Staff

National Defense Hdqtrs, Ottawa
  Logistics, OR Analysis Establishment

American Power Jet Co
  George Chernowitz

General Dynamics, Pomona

General Research Corp
  Dr Hugh Cole
  Library

Logistics Management Institute
  Dr Murray A. Geisler

MATHTEC
  Dr Eliot Feldman

Rand Corporation
  Library

Carnegie-Mellon University
  Dean H. A. Simon
  Prof G. Thompson

Case Western Reserve University
  Prof B. V. Dean
  Prof M. Mesarovic
  Prof S. Zacks

Cornell University
  Prof R. E. Bechhofer
  Prof R. W. Conway
  Prof Andrew Schultz, Jr.

Cowles Foundation for Research in Economics
  Prof Herbert Scarf
  Prof Martin Shubik

Florida State University
  Prof R. A. Bradley

Harvard University
  Prof K. J. Arrow
  Prof W. G. Cochran
  Prof Arthur Schleifer, Jr.

Princeton University
  Prof A. W. Tucker
  Prof J. W. Tukey
  Prof Geoffrey S. Watson

Purdue University
Prof S. S. Gupta
Prof H. Rubin
Prof Andrew Whinston

Stanford University
Prof T. W. Anderson
Prof G. B. Dantzig
Prof F. S. Hillier
Prof D. L. Iglehart
Prof Samuel Karlin
Prof G. J. Lieberman
Prof Herbert Solomon
Prof A. F. Veinott, Jr.

University of California, Berkeley
Prof R. E. Barlow
Prof D. Gale
Prof Jack Kiefer
Prof Rosedith Sitgreaves

University of California, Los Angeles
Prof J. R. Jackson
Prof R. R. O'Neill

University of North Carolina
Prof W. L. Smith
Prof M. R. Leadbetter

University of Pennsylvania
Prof Russell Ackoff
Prof Thomas L. Saaty

University of Texas
Prof A. Charnes

Yale University
Prof F. J. Anscombe
Prof I. R. Savage

Prof Z. W. Birnbaum
University of Washington

Prof B. H. Bissinger
The Pennsylvania State University

Prof Seth Bonder
University of Michigan

Prof G. E. P. Box
University of Wisconsin

Dr Jerome Bracken
Institute for Defense Analyses

Prof H. Chernoff
Mass. Institute of Technology

Prof Arthur Cohen
Rutgers - The State University

Mr Wallace M. Cohen
US General Accounting Office

Prof C. Derman
Columbia University

Prof Masao Fukushima
Kyoto University

Prof Saul I. Gass
University of Maryland

Dr Donald P. Gaver
Carmel, California

Prof Amrit L. Goel
Syracuse University

Prof J. F. Hannan
Michigan State University

Prof H. O. Hartley
Texas A & M Foundation

Mr Gerald F. Hein
NASA, Lewis Research Center

Prof W. M. Hirsch
Courant Institute

Dr Alan J. Hoffman
IBM, Yorktown Heights

Prof John R. Isbell
State University of New York, Amherst

Dr J. L. Jain
University of Delhi

Prof J. H. K. Kao
Polytech Institute of New York

Prof W. Kruskal
University of Chicago

Mr S. Kumar
University of Madras .

Prof C. E. Lemke
Rensselaer Polytech Institute

Prof Loynes
University of Sheffield, England

Prof Steven Nahmias
University of Pittsburgh

Prof D. B. Owen
Southern Methodist University

Prof E. Parzen
Texas A & M University

Prof H. O. Posten
University of Connecticut

Prof R. Remage, Jr.
University of Delaware

Prof Hans Riedwyl
University of Bern

Dr Fred Rigby
Texas Tech College

Mr David Rosenblatt
Washington, D. C.

Prof M. Rosenblatt
University of California, San Diego

Prof Alan J. Rowe
University of Southern California

Prof A. H. Rubenstein
Northwestern University

Dr M. E. Salveson
West Los Angeles

Prof Edward A. Silver
University of Waterloo, Canada

Prof M. J. Sobel
Georgia Inst of Technology

Prof R. M. Thrall
Rice University

Dr S. Vajda
University of Sussex, England

Prof T. M. Whitin
Wesleyan University

Prof Jacob Wolfowitz
University of South Florida

Prof Max A. Woodbury
Duke University

December 1978